Discovering Scientific Relationship Paths using Linked Open Data

Chatzimanolis Charisios

SID: 3301140001

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Information and Communication Systems

OCTOBER 2015

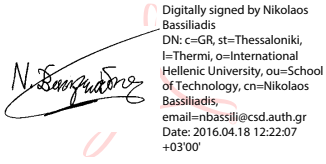
THESSALONIKI – GREECE

Discovering Scientific Relationship Paths using Linked Open Data

Chatzimanolis Charisios

SID: 3301140001

Supervisor:	Assoc. Prof. Nick Bassiliades
Supervising Committee Members:	Dr. V. Peristeras
	Dr. C. Berberidis



SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Information and Communication Systems

OCTOBER 2015

THESSALONIKI – GREECE

Abstract

This dissertation was written as a part of the MSc in ICT Systems at the International Hellenic University.

The application that was implemented is able to find scientific paths between computer scientists' semantic web profiles. It was created in order to highlight the importance of this field and contribute in the next major evolution in connecting information. The fields that where studied for this dissertation were the Semantic Web, the Information Retrieval process, the Open Linked Data as well as the RDF format of data and the SPARQL querying language. It is about a web application where the user gives as an input the names of two computer scientists. A connection with the DBLP data resources allows SPARQL queries to try and retrieve information about the scientific relationships of these two. The level of the relationship between them is ranked in three levels.

At this point I would like to thank my supervisor Dr. Nick Bassiliades for the trust that he showed by assigning me for this dissertation. I would like to thank him for his contribution, the guidance and the assisting bibliography.

Charisios Chatzimanolis

December 11th 2015

Contents

1 Introduction

1.1 DISSERTATION'S SUBJECT

The aim of the thesis is to develop a prototype of an application that will be able to find scientific relationships between researchers through the discovery of RDF graph paths between Linked Open Data of scientific publications. More specifically a combination of information retrieval and semantic web technologies, will give to the users-scientists the ability to establish a link between their semantic web scientific profiles. This assumes that both of them have presence in some rich scientific linked open dataset and that the app has certain patterns of predefined graph traversal pathways (in e.g. SPARQL queries) that tries to find links in the RDF graph between them. Some links could be "the two scientists have presented a paper at the same conference or one of them has cited the paper of the other or they have similar research interests".

Some other uses that this application would be able to provide will be, help for a conference chair or a journal editor to find scientists with specific scientific background to serve in the conference committee or to review a paper. Such a functionality could be to establish a query that returns scientists that have published papers with a topic identical or similar (semantically similar, using an ontology of topics) to the topic in question.

1.2 DISSERTATION'S OBJECTIVE

The objective of this project is the research of the Linked Open Data through a range of technologies such as RDF graphs, SPARQL queries in order to be created an application that will highlight the importance of the Semantic Web. Sir Tim Berners Lee has also pointed the revolution that the linked data could create in the World Wide Web, by sharing a vision of a World Wide Web of data that machines could process independently of humans, enabling a host of new services transforming our everyday lives.

1.3 BACKGROUND

The computer science fields that were studied and researched in order this dissertation to be completed was mainly the Semantic Web as an extension of the Web through standards by the World Wide Web Consortium (W3C), and its principles such as "*Everything can be identified by URI's", "Recourses and Links can have types"* etc. After that, the Linked Open Data and the way that they connect the Web are studied in depth with the help of the RDF Schema formats and the SPARQL querying language. The information retrieval process was major part of the research as well as the Java technologies for the implementation of the application.

1.4 ORIGINAL RESULTS OF THE PROJECT

The project was successfully finished with the creation of the prototype of an application as it was set in the dissertation's goals. This application which was implemented gives the opportunity to the user to realise the importance of the Linked Open Data by providing scientific relationship paths between scientists' semantic web profiles by using RDF graphs. More specifically this tool can find three relational levels which are the following:

1. Closest connection between two scientists is that they have published together an article on a Conference or in a Journal. They are co-authors.
2. The second level of relationship between two scientists is that they had participated in the same conference or journal with their own article or paper.
3. The third level of relationship between the scientists is their background regarding the keywords of their semantic web.

However, despite the reaching of the dissertation's goal, the lack of many Open Data Resources it was an inhibitor for the research of Linked Data in greater range. Which had as a result the application to be used only by one data bank, the DBLP.

2 LITERATURE REVIEW

In this chapter it is described the background and the state of the art of the semantic web as well as of its querying methods. At first the main idea of the semantic web is discussed. Later, in sections 2.2 and 2.3 and 2.4 the information retrieval process, the semantic web problems and the idea of the Linked data and especially the linked open data are defined. The last two sections present in detail the RDF model of the data representation in the web and the SPARQL querying language, which will be used in this master thesis. The final sections briefly present the Apache Jena toolset that will be used for the implementation of the application and the selected programming languages.

2.1 Semantic Web

The World Wide Web contains huge amounts of information created by many different organizations, communities and individuals for many different reasons. Users of the Web can easily access this information by specifying URI addresses, searching, and following links to find other related resources. The simplicity of usage is a key aspect that made Web so popular; so popular in fact, it is hard to imagine life without it anymore.

This simplicity of the current web has a price. It is very easy to get lost, or discover irrelevant or unrelated information with all that is available. The goal of the Semantic Web is to develop enabling standards and technologies designed to help machines understand more information on the Web so that they can support richer discovery, data integration, navigation, and automation of tasks. With Semantic Web we not only receive more exact results when searching for information, but also know when we can integrate information from different sources, know what information to compare, and can provide all kinds of automated services in different domains from future home and digital libraries to electronic business and health services.

More specifically, the Semantic Web is an extension of the Web through standards by the World Wide Web Consortium (W3C). The standards promote common data formats

and exchange protocols on the Web, most fundamentally the Resource Description Framework (RDF).

According to the W3C, "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries". The term was coined by Tim Berners-Lee for a web of data that can be processed by machines. He also indicated that The real power of the Semantic Web will be realized when people create many programs that collect Web content from diverse sources, process the information and exchange the results with other programs. The effectiveness of such software agents will increase exponentially as more machine readable Web content and automated services (including other agents) become available [8].

The main principles [19] of Semantic web are:

1. *Everything can be identified by URI's*: People, places, and things in the physical world can be referred to in the Semantic Web by using a variety of identifiers. Anyone who has control over a part of Web namespace can create a URI and say that it identifies something in the physical world.
2. *Recourses and Links can have types*: The current Web consists of resources and links. The resources are Web documents targeted for human consumption and do not commonly contain metadata explaining what they are used for and what are their relationships to other Web documents. The Semantic Web consists of resources and links. However, now the resources and links can have types which define concepts that tell a bit more to the machines.
3. *Partial information are tolerated*: The Semantic Web is unbounded. Anyone can say anything about anything and create different types of links between resources. There will always be more to discover. Some of the linked resources may cease to exist or the addresses may be reused. The Semantic Web tools need to tolerate this data decay and be able to function in spite of that.
4. *There is no need for absolute truth*: Not everything found from the Web is true and the Semantic Web does not change that in any way. Truth - or more pragmatically, trustworthiness - is evaluated by each application that processes the information on the Web. The applications decide what they trust by using the context of the statements.
5. *Evolution is supported*: The Semantic Web uses descriptive conventions that can expand as human understanding expands. In addition, the conventions allow ef-

fective combination of the independent work of diverse communities even when they use different vocabularies. The Semantic Web provides communities tools that can be used to resolve ambiguities and clarify inconsistencies. Also new information can be added without insisting that the old has to be modified.

6. *Minimalist design*: The Semantic Web makes the simple things simple, and the complex things possible. The aim of the W3C activity is to standardize no more than is necessary. This approach enables the implementation of simple applications now that are based on already standardized technologies.

2.2 The problems of Semantic Web

The Semantic Web allows data to be shared and reused across application, enterprise, and community boundaries. This ability gives the impression that the semantic web's data frames would have been very popular on the web development. However only the 5% of the data in the web are in RDF or any other Ontology format. Most of the people do not know what the word "Semantic" means. The real reason is that the freedom that it is provided from the Web makes it difficult for the structured RDF and OWL formats to be adopted from the users. Some of the open problems of the Semantic Web are:

- How can be build a process model that can make people to adopt the well-defined standards before they publish the data?
- Can a Startup make a dent in this space by thinking of another way to express relationships besides just Triples?
- Can the SILOs of content such as Wikipedia, Twitter, Facebook and LinkedIn be converted into a semantic web format and then to be build a knowledge base?
- How a large Database Product can be created to manage the efficient storage, query and retrieval of Semantic Web data including all representations and interlinking?

Summarizing, the problem of the semantic web is that the visions of how it will applied is lacking. A very small percentage of organizations are using semantic web standards in a highly effective way, while most of the rest are struggling with how to apply them

at all or suffer from ignorance or indifference to them. A few are systematically trying to apply the technology effectively, but haven't succeeded yet. Search engine companies suffer from a mindset that embraces legacy search and retrieval, rather than new forms of graph-based browsing and discovery that are just emerging.

2.3 Information Retrieval

Information retrieval (IR) is the activity of obtaining information resources relevant to an information need from a collection of information resources. Searches can be based on metadata or on full-text (or other content-based) indexing. Web search engines are the most visible IR applications.[1]

The term IR was introduced by Calvin Mooers in 1951, who defined it in this way:

"Information retrieval is the name for the process or method whereby a prospective user of information is able to convert his need for information into an actual list of citations to documents in storage containing information useful to him. It is the finding or discovery process with respect to stored information. It is another, more general, name for the production of a demand bibliography. Information retrieval embraces the intellectual aspects of the description of information and its specification for search, and also whatever systems, technique, or machines that are employed to carry out the operation. Information retrieval is crucial to documentation and organization of knowledge". (Mooers, 1951, p. 25). [2]

The modern Information Retrieval process includes Modeling, Web Search, text classification, data visualization and filtering. The World Wide Web has given a new dimension on the idea of information retrieval due to great amount of structured information that are “stored” in the web. An information retrieval process begins when a user enters a query into the system. The process is presented in Figure 2.1.1 . Queries are formal statements of information needs, for example search strings in web search engines. In information retrieval a query does not uniquely identify a single object in the collection. Instead, several objects may match the query, perhaps with different degrees of relevancy.

An object is an entity that is represented by information in a database. User queries are matched against the database information. Depending on the application the data objects may be, for example, text documents, images, audio, mind maps or videos. Often the

documents themselves are not kept or stored directly in the IR system, but are instead represented in the system by document surrogates or metadata.

Most IR systems compute a numeric score on how well each object in the database matches the query, and rank the objects according to this value. The top ranking objects are then shown to the user. The process may then be iterated if the user wishes to refine the query.[3]

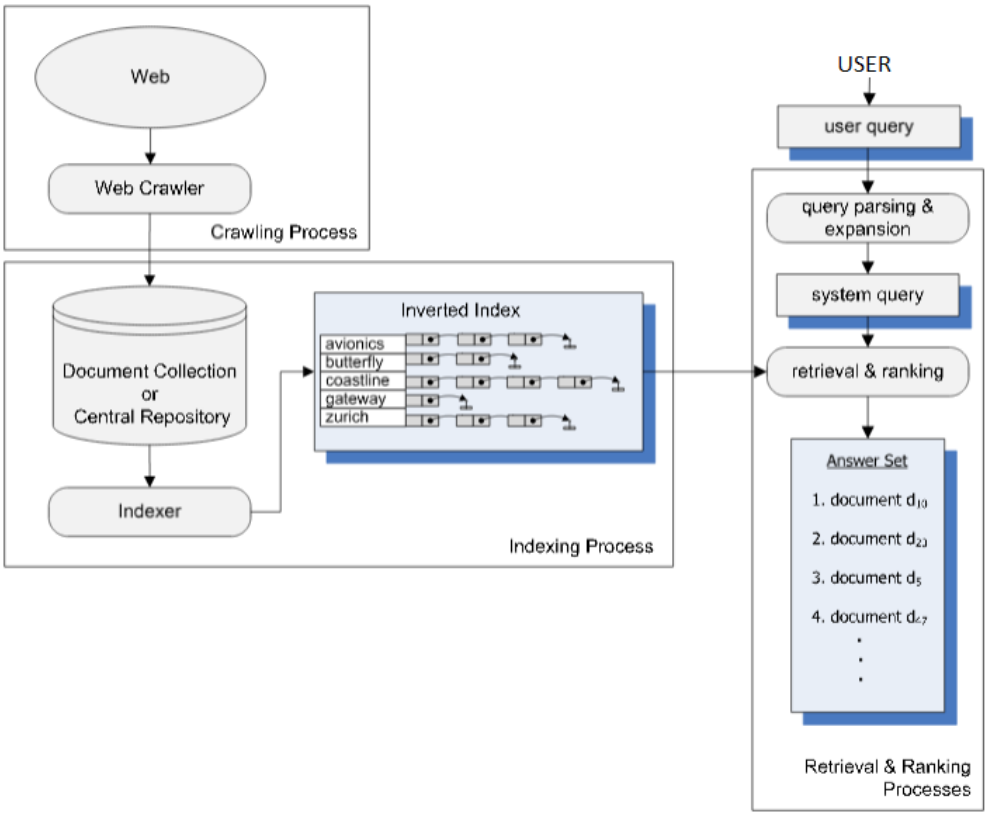


Figure 2.3.1 The Information Retrieval process [4]

The process of the information retrieval will be applied in this dissertation project in order to be achieved the extraction of the best results. The user's input will be combined with a number of different SPARQL queries that will retrieve data from the web and more specifically from the Central Repository of DBLP where the data are stored in RDF format. Depending on the SPARQL query, the results will be ranked relying on the similarity criteria that will be added.

2.4 Linked Data

2.4.1 Linked Data

The Semantic Web is a Web of Data — of dates and titles and part numbers and chemical properties and any other data one might conceive of. The collection of Semantic Web technologies (RDF, OWL, SKOS, SPARQL, etc.) provides an environment where application can query that data, draw inferences using vocabularies, etc.

However, to make the Web of Data a reality, it is important to have the huge amount of data on the Web available in a standard format, reachable and manageable by Semantic Web tools. Furthermore, not only does the Semantic Web need access to data, but relationships among data should be made available, too, to create a Web of Data (as opposed to a sheer collection of datasets). This collection of interrelated datasets on the Web can also be referred to as Linked Data.

To achieve and create Linked Data, technologies should be available for a common format (RDF), to make either conversion or on-the-fly access to existing databases (relational, XML, etc). It is also important to be able to setup query endpoints to access that data more conveniently [5].

In summary [20], Linked Data is simply about using the Web to create typed links between data from different sources. These may be as diverse as databases maintained by two organizations in different geographical locations, or simply heterogeneous systems within one organization that, historically, have not easily interoperated at the data level.

Technically, Linked Data refers to data published on the Web in such a way that it is machine-readable, its meaning is explicitly defined, it is linked to other external data sets, and can in turn be linked to from external data sets. While the primary units of the hypertext Web are HTML (HyperText Markup Language) documents connected by untyped hyperlinks, Linked Data relies on documents containing data in RDF (Resource Description Framework) format (Klyne and Carroll, 2004). However, rather than simply connecting these documents, Linked Data uses RDF to make typed statements that link arbitrary things in the world. The result, which we will refer to as the Web of Data, may more accurately be described as *a web of things in the world, described by data on the Web.*

2.4.2 Linked Open Data

The most visible example of adoption and application of the Linked Data principles has been the Linking Open Data project.

Linked Open Data are Open Data that must be linked, otherwise are just Open Data. Some of the biggest Open Linked Data are the DBpedia and Freebase. Tim Berners-Lee[6] gives the clearest definition of linked open data in differentiation with linked data. He defines linked data by identifying its four components, Machine Readable, Non-proprietary Format, RDF Standards, Linked RDF, and then adds a fifth rule, open in the web with an open license. In other words the open linked data can be described from the five-star open linked data schema.

*	Available on the web (whatever format) *but with an open license, to be Open Data*
* *	Available as machine-readable structured data (e.g. excel instead of image scan of a table)
* * *	as (2) plus non-proprietary format (e.g. CSV instead of excel)
* * * *	All the above plus, Use open standards from W3C (RDF and SPARQL) to identify things, so that people can point at your stuff
* * * * *	All the above, plus: Link your data to other people's data to provide context

Figure 2.4.1 Five Star Open Linked Data Schema.

The Linked Open Data[20] project is a a grassroots community effort founded in January 2007 and supported by the W3C Semantic Web Education and Outreach Group. The original and ongoing aim of the project is to bootstrap the Web of Data by identifying existing data sets that are available under open licenses, converting these to RDF according to the Linked Data principles, and publishing them on the Web. Participants in the early stages of the project were primarily researchers and developers in university research labs and small companies. Since that time the project has grown considerably, to include significant involvement from large organisations such as the BBC, Thomson Reuters and the Library of Congress. This growth is enabled by the open

nature of the project, where anyone can participate simply by publishing a data set according to the Linked Data principles and interlinking it with existing data sets. An indication of the range and scale of the Web of Data originating from the Linking Open Data project is provided in Figure 2.3.1. Each node in this cloud diagram represents a distinct data set published as Linked Data, as of March 2009.

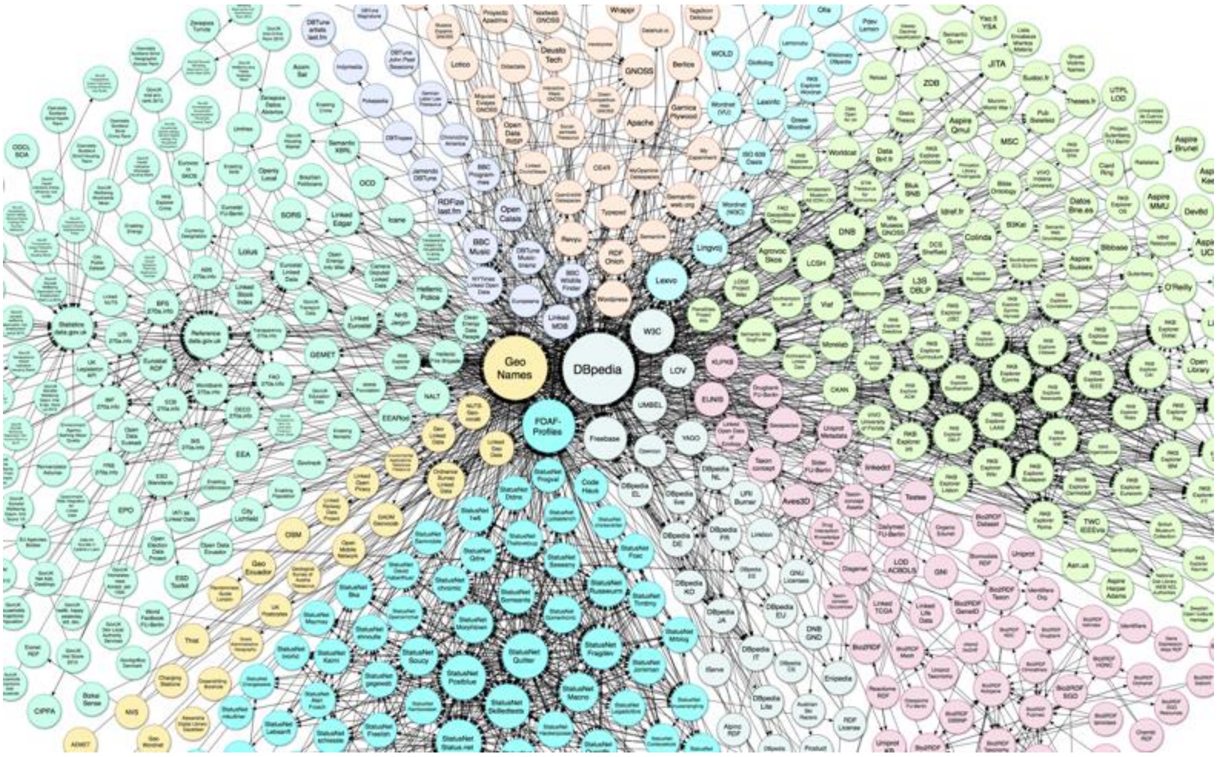


Figure 2.4.2 World Wide Web's Linked Data[7].

2.5 RDF

"The Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model. It has come to be used as a general method for conceptual description or modelling of information that is implemented in web resources, using a variety of syntax notations and data serialization formats. It is also used in knowledge management applications.

The RDF data model[10] is similar to classical conceptual modeling approaches such as entity–relationship or class diagrams, as it is based upon the idea of making statements about resources (in particular web resources) in the form of subject–predicate–object expressions. These expressions are known as *triples* in RDF terminology. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object. For example, one way to represent the notion "The sky has the color blue" in RDF is as the triple: a subject denoting "the sky", a predicate denoting "has", and an object denoting "the color blue". Therefore, RDF swaps object for subject that would be used in the classical notation of an entity–attribute–value model within object-oriented design; object (sky), attribute (color) and value (blue). RDF is an abstract model with several serialization formats (i.e., file formats), and so the particular way in which a resource or triple is encoded varies from format to format."

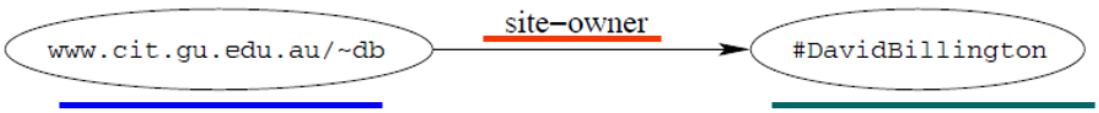


Figure 2.5.1 A graphical representation of triples.

The previews figure shows the blue the subject. The red is used for the attribute and the green is for the object. This mechanism for describing resources is a major component in the W3C's Semantic Web activity: an evolutionary stage of the World Wide Web in which automated software can store, exchange, and use machine-readable information distributed throughout the Web, in turn enabling users to deal with the information with greater efficiency and certainty. RDF's simple data model and ability to model disparate, abstract concepts has also led to its increasing use in knowledge management applications unrelated to Semantic Web activity.

A collection of RDF statements intrinsically represents a labeled, directed multi-graph. As such, an RDF-based data model is more naturally suited to certain kinds of knowledge representation than the relational model and other ontological models. However, in practice, RDF data is often persisted in relational database or native representations also called Triplestores, or Quad stores if context (i.e. the named graph) is also persisted for each RDF triple[9]. ShEX, or Shape Expressions, is a language for expressing constraints on RDF graphs. It includes the cardinality constraints from OSLC Re-

source Shapes and Dublin Core Description Set Profiles as well as logical connectives for disjunction and polymorphism. As RDFS and OWL demonstrate, one can build additional ontology languages upon RDF.

2.6 Querying Languages

2.6.1 SPARQL

"SPARQL (Protocol and RDF Query Language) is an RDF query language, that is, a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format. [11][12] It was made a standard by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium, and is recognized as one of the key technologies of the semantic web. On 15 January 2008, SPARQL 1.0 became an official W3C Recommendation and SPARQL 1.1 in March, 2013.[35]"

SPARQL allows for a query to consist of triple patterns, conjunctions, disjunctions, and optional patterns. More Specifically [21], SPARQL is essentially a graph-matching query language. An example can be seen in the following Figure where different information about DBLP and DBpedia are stored. There can be seen the RDF triplets which consist of two a bullets and a line that connect them, and the prefixes, foaf, rdfs, dbpedia, owl stc.

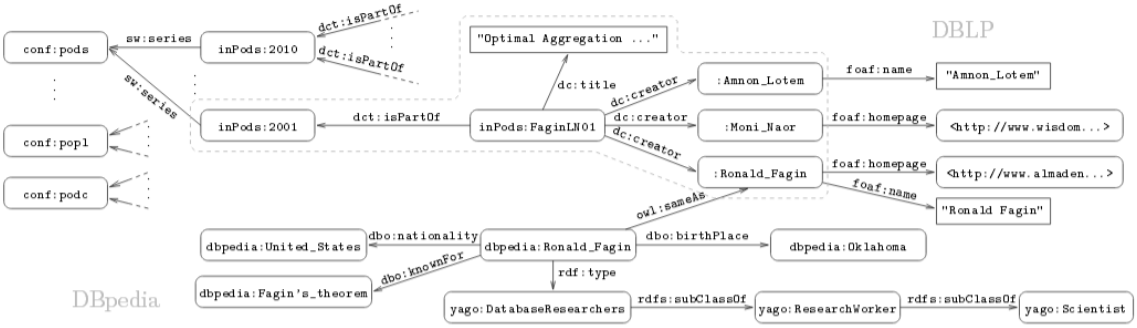


Figure 2.6.1 An RDF graph storing DBLP and DBpedia information

A SPARQL query is composed of: (1) a body, which is a complex RDF graph pattern-matching expression, and (2) a head, which is an expression that indicates how to construct the answer to the query. The following is an example of a simple SPARQL query that retrieves the authors that have published in PODS and were born in Oklahoma, following the RDF graph from above. The body of the query is just a set of triples with

variables, which are prefixed by the symbol ?, and the head is a projection (via a SELECT operator):

SELECT ?Author

WHERE {

?Paper dc:creator ?Author .

?Paper dct:isPartOf ?InPods .

?InPods sw:series conf:pods .

?DbpPerson owl:sameAs ?Author .

?DbpPerson dbo:birthPlace dbpedia:Oklahoma . }

The evaluation of a query as the one shown above is done in two steps. First, values are given to variables in such a way that the triples in the body of the query match the triples in the queried graph (Figure 2.6.1.1). Second, the values assigned to the variables are processed to construct the answer. In this case, some variables are projected out to keep only the author. In this example Ronald_Fagin is an answer to this query.

There exist tools that allow one to connect and semi-automatically construct a SPARQL query for a SPARQL endpoint, for example ViziQuer where the user can connect to a SPARQL endpoint and construct simple graphical queries to retrieve data. ViziQuer also allows the exploration of SPARQL endpoint data schema. It is designed to easily define restricted data set for further analysis or just data browsing.

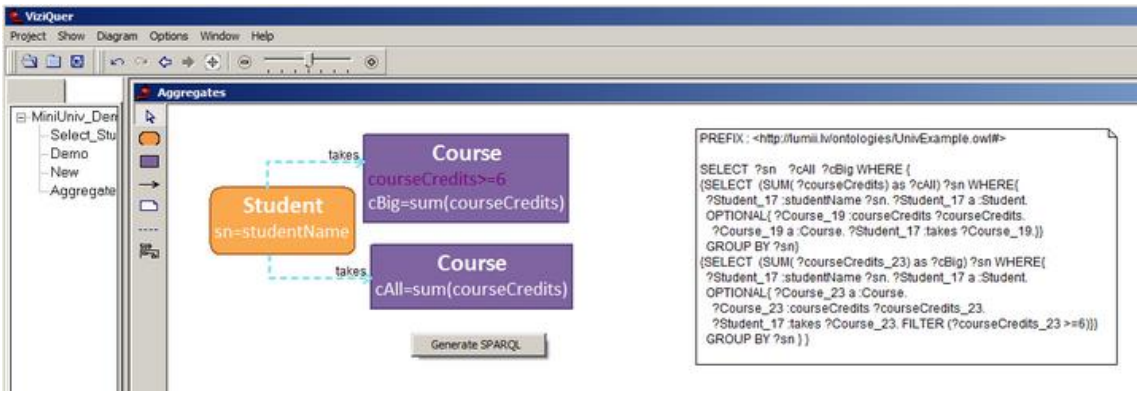


Figure 2.6.2 ViziQuer auto generation of SPARQL queries.

In addition, there are tools that translate SPARQL queries to other query languages, for example to SQL and to XQuery.

SPARQL allows users to write queries against data that can loosely be called "key-value" data or, more specifically, data that follows the RDF specification of the W3C. The entire database is thus a set of "subject-predicate-object" triples. This is analogous to some NoSQL databases' usage of the term "document-key-value", such as MongoDB.

RDF data can also be considered in SQL relational database terms as a table with three columns – the subject column, the predicate column and the object column. Unlike relational databases, the object column is heterogeneous, the per-cell data type is usually implied (or specified in the ontology) by the predicate value. An example is shown in the following Figure.

Triples of the Data Model

Number	Subject	Predicate	Object
1	http://www.mydomain.org/uni-ns#T949318	http://www.mydomain.org/uni-ns#name	"David Billington"
2	http://www.mydomain.org/uni-ns#T949318	http://www.mydomain.org/uni-ns#title	"Associate Professor"
3	http://www.mydomain.org/uni-ns#T949318	http://www.mydomain.org/uni-ns#age	"27"^^http://www.w3.org/2001/XMLSchema#integer
4	http://www.mydomain.org/uni-ns#CIT1111	http://www.mydomain.org/uni-ns#courseName	"Discrete Maths"
5	http://www.mydomain.org/uni-ns#CIT1111	http://www.mydomain.org/uni-ns#isTaughtBy	http://www.mydomain.org/uni-ns#T949318

Figure 2.6.3 Data Table of RDF triplets

Alternately, again comparing to SQL relational, all of the triples for a given subject could be represented as a row, with the subject being the primary key and each possible predicate being a column and the object is the value in the cell. However, SPARQL/RDF becomes easier and more powerful for columns that could contain multiple values (like "children"), and where the column itself could be a joinable variable in the query, rather than directly specified. The following Figure provides an explanation of the procedure that a Relational database can be converted into RDF schema.

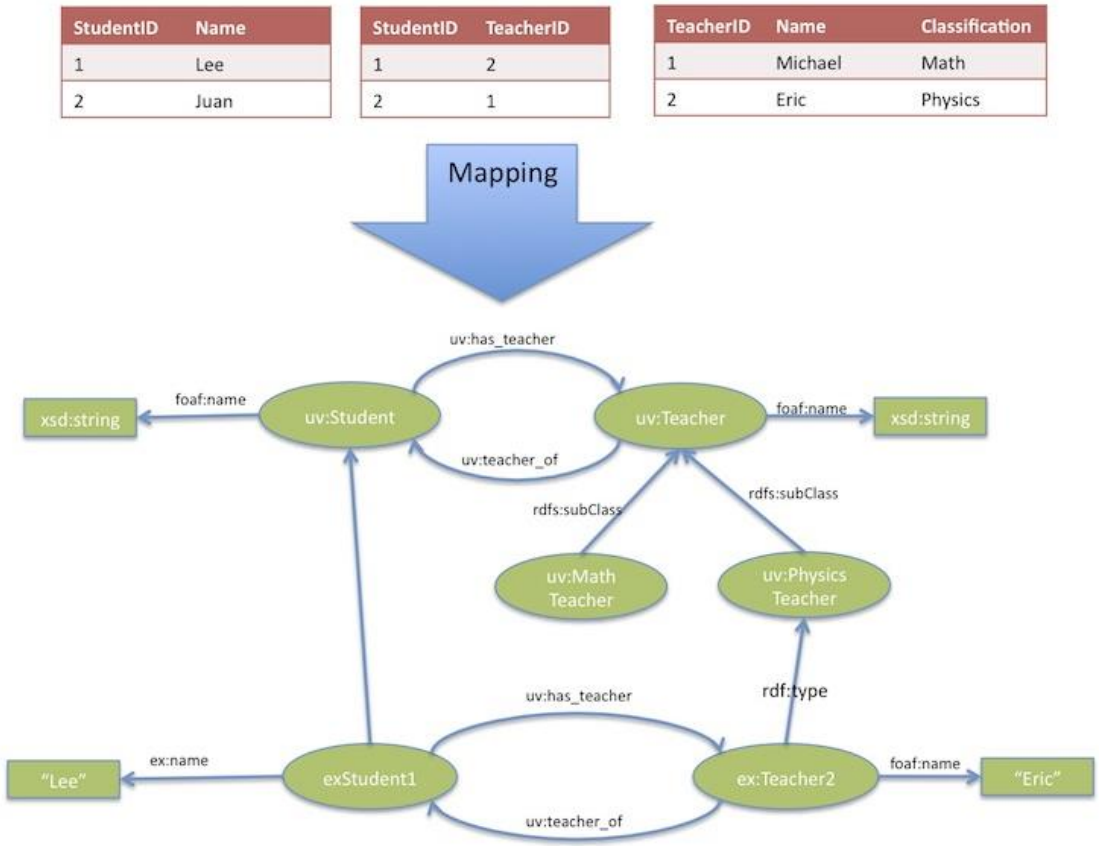


Figure 2.6.4 SQL data tables mapping to RDF graph schema

SPARQL thus provides a full set of analytic query operations such as JOIN, SORT, AGGREGATE for data whose schema is intrinsically part of the data rather than requiring a separate schema definition. Schema information (the ontology) is often provided externally, though, to allow different datasets to be joined in an unambiguous manner. In addition, SPARQL provides specific graph traversal syntax for data that can be thought of as a graph.

2.6.2 SNORQL

SNORQL is an AJAX front-end for exploring RDF SPARQL endpoints. It was originally created by Richard Cyganiak[14] for the D2R server project[15].

SNORQL can be used with just about any SPARQL endpoint that supports JSON. The syntax and the prefixes of SNORQL are the same as SPARQL, but the first provides the ability to the user to see if his query works, live and navigate his linked data. To resume SNORQL is used for querying semantic web via web browsers interface as a search engine and the result are represented in html format. Unlike SNORQL, SPARQL's end-

point cannot be presented as a querying interface. The following Figure shows the SNORQL endpoint interface of d2r server for the dblp data resource.

Figure 2.6.5 The Snorql endpoint of DBLP on d2r Server

The specific interface allows the user to query the dblp and find scientific information, such as conferences, publications on magazines, authors of the published articles etc, about Computer Science. The 10 Prefixes are standard and referred to dblp. The white box is used from the user to write his query in SPARQL. He has also the ability to add more prefixes in order to access the information that he needs easier. Finally the results can be presented in HTML format, JASON, XML and XML + XSLT.

2.7 Apache Jena

"Apache Jena[13] is an open source Semantic Web framework for Java. It provides an API to extract data from and write to RDF graphs. The graphs are represented as an abstract "model". A model can be sourced with data from files, databases, URLs or a combination of these. A Model can also be queried through SPARQL 1.1.

Jena is similar to Sesame; though, unlike Sesame, Jena provides support for OWL (Web Ontology Language)." The framework has various internal reasoners and the Pellet rea-

soner (an open source Java OWL-DL reasoner) can be set up to work in Jena. Regarding the RDF. Jena provides an RDF API. Interact with the core API to create and read Resource Description Framework (RDF) graphs, and serialize the triples using popular formats such as RDF/XML or Turtle. For querying the triplets Jena also contain a SPARQL 1.1 compliant engine. ARQ supports also remote federated queries and free text search. After these, Jena can Persist users' data using TDB, a native high performance triple store. TDB supports the full range of Jena APIs. Another ability is the Exposure of the triples as a SPARQL end-point accessible over HTTP by Fuseki. It also provides REST-style interaction with the RDF data. Finally, Jena is compatible with the OWL format. It works with models, RDFS and the Web Ontology Language (OWL) to add extra semantics to your RDF data. Regarding the Inference API, it gives the user the opportunity to reason over his data to expand and check the content of his triple store. Configure his own inference rules or use the built-in OWL and RDFS reasoners.

2.8 Data Accessing

The goal of this project is to find scientific relationship paths between researchers or articles from some scientific magazines or conferences. There are two different techniques to implement the tool. The first one is to connect directly to server that gives access to the data sources. The second requires the data to be stored in a local machine and be used locally.

2.8.1 Connecting to Server

There are organizations that give access to their data resources. Everyone can access the information that he is interested in by connecting to their server and by using their own endpoint to query the information. The specific project needed access to a database that contains data in RDF format. There was a challenge in finding these kind of data resources and the server that can offer this ability. Solution in this problem could have been given from the D2RQ Server. As it is analysed later, D2RQ is a project that was created for handling traditional database systems and query them as if they were RDF data with SPARQL queries. This project was adopted from several organizations who want to create an online endpoint for the users to query their Relational Data via a semantic view.

2.8.2 Local Storage of Data

The easiest technique for handling data is to have them stored locally. This gives the ability to the administrator to have full control over all the data and their format and find the best ways to manipulate them. This technique could be the best for the creation of the tool with freedom. It gives also the power to the administrator to change the format of the data and in general to handle them in the most efficient way.

Programming Language

2.9 Programming Language

This Chapter presents all the available programming languages with their features and advantages or disadvantages for the implementation of the tool for the dissertation project.

2.9.1 Java

Java is a programming language expressly designed for use in the distributed environment of the Internet. It was designed to have the "look and feel" of the C++ language, but it is simpler to use than C++ and enforces an object-oriented programming model. Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network. It can also be used to build a small application module or applet for use as part of a Web page. Applets make it possible for a Web page user to interact with the page.

The source program is compiled into what Java calls bytecode, which can be run anywhere in a network on a server or client that has a Java virtual machine.

Java has also the ability to use a huge number of internal or external libraries and add new features and abilities. For example the Apache Jena [28] gives several libraries for the manipulation of data in RDF format which it would be a great feature for the implementation of the tool for the specific project. More specifically it provides an API to extract data from and write to RDF graphs. The graphs are represented as an abstract "model". A model can be sourced with data from files, databases, URLs or a combination of these. A Model can also be queried through SPARQL 1.1.

2.9.2 C#

C# is an object-oriented programming language from Microsoft that aims to combine the computing power of C++ with the programming ease of Visual Basic. C# is based on C++ and contains features similar to those of Java. C# is designed to work with Microsoft's .Net platform.

C# enables developers to build highly portable applications. It also simplifies programming through its use of Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP) which allow access to a programming object or method without requiring the programmer to write additional code for each step.

Regarding its ability to work with RDF data there are several libraries such as dotNetRDF[29] open source C# .NET library for RDF which also support Apache Jena Fuseki and other APIs for RDF and OWL or SPARQL. Another one interesting framework is RDFSharp[30] which is a lightweight C# framework designed to ease the creation of .NET applications based on the RDF model, representing a straightforward didactic solution for start playing with RDF and Semantic Web concepts.

With RDFSharp it is possible to realize .NET applications capable of modeling, storing and querying RDF data.

2.9.3 Comparison

Comparing the programming languages Java and C#, both of them are simple to use, easy to create small applications for the web and also both of them can use external libraries or work frames for handling RDF data. C# seems to have more features based on RDF or SPARQL while Java has more advantages and features on creating web application (Aplets).

The following table contains some of the frameworks that can be used from Java and C# to handle data of the Semantic Web in RDF format.

Java	*C#*
Corese	RDFSharp
DataGrid	DotSesame
Jena	Drive
JRDF	SemWeb

RDFSuite	dotNetRdf
Sesame	SesameWindowsClient

Table 2.9.1 presentation of frameworks for using RDF data in Java and C#

2.10 Conclusion

In this chapter were mentioned which are exactly the main problems of the semantic web, for finding paths between open linked data. There were presented some definitions and it was described the way that the data can be structured in the web by following the RDF format, and get queried with SPARQL. Finally, there was a brief description of the technologies around the semantic web and the tools that can be used for manipulation of these data.

3 DESIGNING THE SOLUTION

In this chapter the procedure of the research is presented for the choice of the best and more suitable data base with linked data that could be used for the implementation of the application. It is analyzed the choice of the right one and the components that made it suitable for finding relational paths among the data that contains. There are discussed also some additional decisions for the implementation of the final application. Finally, are addressed the problems that should be studied in order the dissertation to be successfully finished.

3.1 Addressing the Problem

It was presented before that the aim of the thesis is to develop a prototype of an application that will be able to find scientific relationships between researchers through the discovery of RDF graph paths between Linked Open Data of scientific publications. More specifically there is a number of Databases in the web that contain metadata of scientific publications and information for computer scientists. In addition different data bases could possibly contain different kind of information. An example is that a database like DBLP could contain all the metadata for a scientist's publications, books and articles in journals that he has published but do not contain personal information such as his CV, or his career or more personal details that could exist in a database like the DBpedia. So, The first and most important problem is that these Databases are not connected.

Another one, more general problem has to do with how these data can be presented in the web in a unified format in order to be recognised from everyone. In other words what is the format that the data should have in order first of all to be semantic and secondly to be able to be linked? The reason is that this thesis is based on this sector of computer science, and tries to find out how the Linked data can be reclaimed.

The third important problem that needed to be studied was the ability for someone to find these data free-Open in the web. This is because the aim of the dissertation is to be

created a tool that it will be based and it will highlights the importance of free Data on the web which will be accessible from everyone.

Regarding the most specific objectives of the dissertation, the problem had to do with the ability to find relational scientific paths between computer scientists. It was decided that there will be three relational levels. In the first, the scientists will be coauthors of at least one publication. The second relational level it will points if two scientists had joined the same conferences by presenting their own publication. The third and final relational level it will compare the background of the scientists based on the keywords that they use in their semantic profile and finds the similarities. The solution of these problems could be found in queries that use querying language such as sparql. In order to be clearer the scientific relations are based on the right queries that will be able to mine the right results. Finally the last problem that should be solved was the ability to present the data graphically in order to be easier for the user to understand the relations between the scientists.

The dissertation's subject points out that some other uses that this application would be able to provide will be, help for a conference chair or a journal editor to find scientists with specific scientific background to serve in the conference committee or to review a paper. Such a functionality could be to establish a query that returns scientists that have published papers with a topic identical or similar (semantically similar, using an ontology of topics) to the topic in question. This additional problem again it will be based on the right query that will retrieve scientists with semantically similar topics.

The following sub-chapters describe all the efforts, the planning and the solutions on these problems.

3.2 Choosing the Data Sources

There are quite a few Databases that contain linked data in the semantic web form that could be used for finding the required data for this project. As already discussed, the goal of this dissertation is to create a prototype application that will be able to find scientific relationships between semantic web profiles of scientists, and/or between scientific publication fora (conferences, journals, etc.) and scientist profiles. As a result the availability of the semantic data sources was limited by the following basic principles:

- The Data should be in RDF/OWL format so that they will be available for querying via SPARQL or a similar semantic web querying language.
- The Data should contain scientific information for Professors, Scientists, Scientific Conferences, Articles or Magazines. In this project we have chosen to focus on the Computer Science discipline.
- The Database should have an endpoint (i.e. a web API) in order to be able to query the data remotely from a client.
- The data should be Open-free to use.

Following the above principles these are the Data sources that were considered:

1. *ACM Digital Library*.[22] "The ACM Digital Library is the full-text collection of all articles published by the ACM in its articles, magazines and conference proceedings. The Guide is a bibliography in computing with over one million entries. The ACM Digital Library contains a comprehensive archive starting in the 1950s of the organization's journals, magazines, newsletters and conference proceedings. Online services include a forum called Ubiquity and Tech News digest. There is an extensive underlying bibliographic database containing key works of all genres from all major publishers of computing literature. This secondary database is a rich discovery service known as The ACM Guide to Computing Literature. ACM adopted a hybrid OA publishing model in 2013. Authors who do not choose to pay the OA fee must grant ACM publishing rights by either a copyright transfer agreement or a publishing license agreement. See.[16]ACM was a "green" publisher before the term was invented. Authors may post documents on their own websites and in their institutional repositories with a link back to the ACM Digital Library's permanently maintained Version of Record. All metadata in the Digital Library is open to the world, including abstracts, linked references and citing works, citation and usage statistics, as well as all functionality and services. Other than the free articles, the full-texts are accessed by subscription".

2. *Scopus*[23] "is a bibliographic database containing abstracts and citations for academic journal articles. It covers nearly 22,000 titles from over 5,000 publishers, of which 20,000 are peer-reviewed journals in the scientific, technical, medical, and social sciences (including arts and humanities). It is owned by Elsevier and is available online by subscription. Searches in Scopus also incorporate searches of patent databases.[17] Scopus also offers author profiles which cover affiliations, number of publications and their bibliographic data, references, and details on the number of citations each published document has received. It has alerting features that allows registered users to track changes to a profile and a facility to calculate authors' h-index. Scopus can be integrated with ORCID".

3. *DBLP*[24]. "DBLP is a computer science bibliography website hosted at Universität Trier, in Germany. It was originally a database and logic programming bibliography site, and has existed at least since the 1980s. DBLP listed more than 2.9 million journal articles, conference papers, and other publications on computer science in May 2015, up from about 14,000 in 1995. All important journals on computer science are tracked. Proceedings papers of many conferences are also tracked. It is mirrored at five sites across the Internet."

4. *arXiv*[25]. "is a repository of electronic preprints, known as e-prints, of scientific papers in the fields of mathematics, physics, astronomy, computer science, quantitative biology, statistics, and quantitative finance, which can be accessed online. In many fields of mathematics and physics, almost all scientific papers are self-archived on the arXiv. Begun on August 14, 1991, arXiv.org passed the half-million article milestone on October 3, 2008.By 2014 the submission rate had grown to more than 8,000 per month."

5. *Springer*[26] "is a global publishing company that publishes books, e-books and peer-reviewed journals in science, technical and medical (STM) publishing. Springer also hosts a number of scientific databases, including SpringerLink, Springer Protocols, and SpringerImages. Book publications include major reference works, textbooks, monographs and book series; more than 168,000 titles are available as e-books in 24 subject collections."

6. *Semantic Web Dog Food*[27] provides information about conferences, people who attended them, authors and publications related to Semantic Web. It is also sponsored by Semantic Web Science Association (SWSA), NEPOMUK project and DERI, National University of Ireland.

3.3 DBLP

After a thorough examination of the available data sources, DBLP was selected as the best choice for the information retrieval process for the implementation of the application because it was following all of the four principles that had been presented in the chapter 4.1. As it is presented in the Figure 4.2.1 the data are scientific and more specifically from computer science, they have RDF format, the whole database is available for the user to download in RDF/XML format or to query through the SPARQL endpoint on tier2 server, and all data is open-free. Semantic Web Dog Food also covers all these principles but has much less data than DBLP.

	RDF/OWL Format	Computer Science	Endpoint (SPARQL)	Open – Free
ACM		٧		٧
Scopus		٧		٧ (User need a key- free of charge)
DBLP	٧	٧	٧	٧
arXiv		٧		٧
Semantic Web Dog Food	٧	٧	٧	٧ (Semantic web

				only)
Springer		Y		Y

Figure 3.3.1 Table with the followed principles from each Data Source

DBLP, provides open bibliographic information and more specifically metadata on major computer science journals and proceedings. Regarding the way that the DBLP website is organized, there are four different kind of web pages:

- *Index pages*: They contain an alphabetical hierarchy of the conferences and the journals as it can be seen in figure 4.2.1.

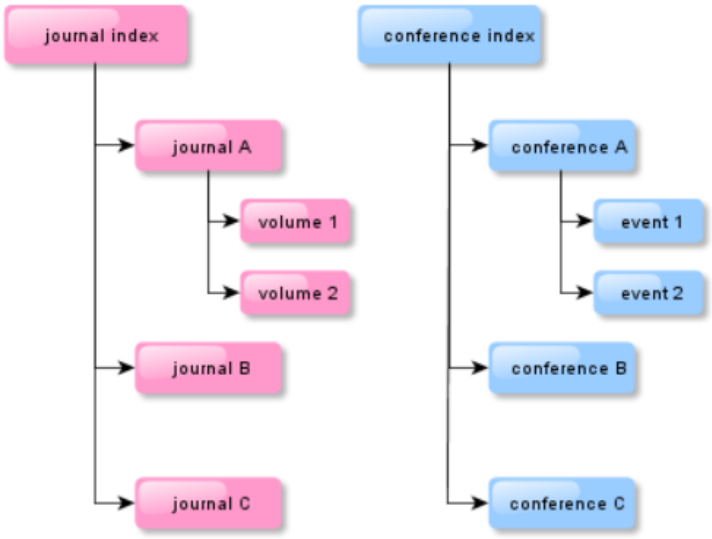


Figure 3.3.2 Journal and Conference indexes.

- *Publication Streams*: They contain links to the tables of contents of the proceedings or journal volumes. Bibliographic information on the proceedings, information on upcoming events and pointers to Web pages of the publishers may also be integrated into a publication stream's page.
- *Table of Context*: These pages list the bibliographic metadata for each article of the volume, that is, the full list of authors, the title, the page numbers, and any available hyperlinks.

- *Authors*: Whenever a person's name appears on the *DBLP* web site, that name provides a hyperlink to the author page of that person. Here, all identified publications of this person are listed.

3.3.1 DBLP Statistics

This subsection contains some very important statistics about the DBLP regarding the number of the publication types and the distribution of the publications regarding their types. The following graphs give a presentation of these kind of information where can be easily seen the amount of information that DBLP contains.

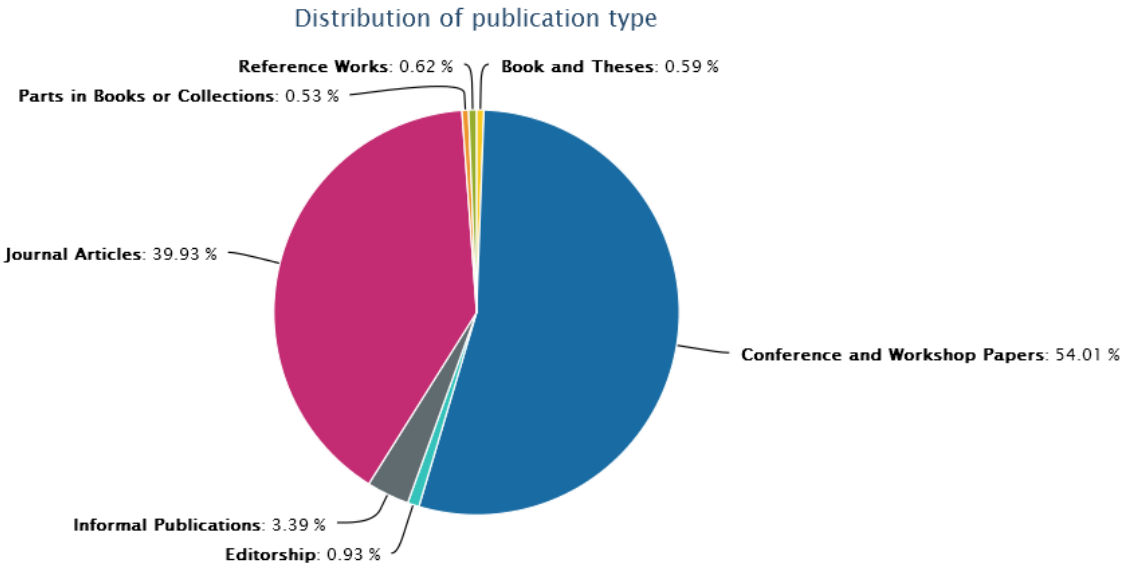


Figure 3.3.3 Distribution of publication type in DBLP

In the previous graph it can be seen that the 54% of the publication data have to do with the Conferences and Workshop papers while the 40% contains data from Journal articles and the other 6% percent of the data consists of Book and Theses, Informal publications and Editorships.

The following graph provide information about the rise of the number of publications that DBLP contains between 1995 and 2015. It is notable that from 2010 there was a sharp rise in the number of publications that has reached 3 million in 2015.

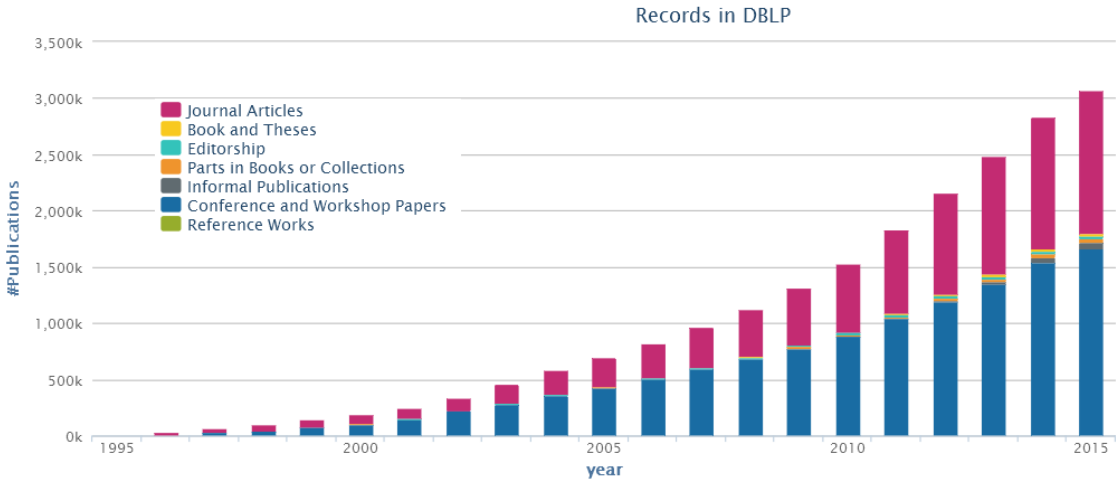


Figure 3.3.4 Records in DBLP

3.3.2 DBLP Analysis

DBLP structure consist of a number of Classes and Properties that can be used for structuring the information in RDF format that can make the metadata accessible easily by using SPARQL queries.

To begin with, there are a couple of standard prefixes or URI's for RDF format that are used from dblp:

- *foaf*: "Is a machine-readable ontology describing persons, their activities and their relations to other people and objects. Anyone can use FOAF to describe him- or herself. FOAF allows groups of people to describe social networks without the need for a centralised database. It is a descriptive vocabulary expressed using the Resource Description Framework (RDF) and the Web Ontology Language (OWL)[36]"
- *swrc*: The SWRC ontology generically models key entities relevant for typical research communities and the relations between them. It is descriptive vocabulary that is used for RDF and OWL. SWRC comprises at total of six top level concepts, namely the Person, Publication, Event, Organization, Topic and Project concepts.
- *owl*: The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting infor-

mation to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics.

- *rdfs*: Is extending RDF vocabulary to allow describing taxonomies of classes and properties. It also extends definitions for some of the elements of RDF, for example it sets the domain and range of properties and relates the RDF classes and properties into taxonomies using the RDFS vocabulary. Some of the classes that are defined by rdfs are Datatypes, subClassOf and some of the defined properties are range, label, value etc.
- *dc/dcterms*: The Dublin Core ontology is a light weight RDFS vocabulary for describing generic metadata. It defines a set of metadata elements for cataloging library items and other electronic resources. Some of its elements are title, creator, publisher, date, format etc.

The following table shows all the Classes that are used for describing the data in DBLP, their properties and what they represent. There are 6 different Classes.

Classes	Contains	Instance	Instances' Properties
swrc:Article	Contains all the instances of articles that have been published in journals, with the name of the journals, articles and the name of the author for each one.	<http://dblp.l3s.de/d2r/resource/publications/journals/4or/Alvarez-Miranda15>	dcterms:bibliographicCitation dc:creator foaf:homepage dc:identifier dcterms:issued swrc:journal rdfs:label foaf:maker swrc:number swrc:pages

			owl:sameAs rdfs:seeAlso dc:title dc:type rdf:type swrc:volume
swrc:Collection	Contains all the collections of Books, by giving information about the book and the author of it.	<http://dblp.l3s.de/d2r/resource/collections/mit/papazoglouST2000>	rdfs:label owl:sameAs is swrc:series of dc:title rdf:type
swrc: Conference	Contains all the main conferences. Each one contains instances with all the available conference.	<http://dblp.l3s.de/d2r/resource/conferences/aaai>	rdfs:label rdfs:seeAlso is swrc:series of dc:title rdf:type
swrc: Journal	Contains all the instances of the main Journals. Each one contains all the published articles.	<http://dblp.l3s.de/d2r/resource/journals/4or>	is swrc:journal of rdfs:label rdfs:seeAlso dc:title rdf:type
foaf: Agent	Contains all the names of the authors of published articles. Each one contains all the articles that he owns.	<http://dblp.l3s.de/d2r/resource/authors/A-Nasser_Ansari>	is dc:creator of rdfs:label is foaf:maker of foaf:name rdfs:seeAlso rdf:type
foaf: Document	Contains all the documents-publications from	<http://dblp.l3s.de/d2r/resource/publicati	dcterms:bibliographicCitation

	journals and conferences	ons/journals/ac/StojanovicBB15>	dc:creator foaf:homepage dc:identifier dcterms:issued swrc:journal rdfs:label foaf:maker swrc:number swrc:pages owl:sameAs rdfs:seeAlso dc:title dc:type rdf:type swrc:volume

Figure 3.3.5 Analysis of the Classes of DBLP

The next table presents all the properties of DBLP and their description.

Properties	Description
dc:creator	An entity primarily responsible for making the resource. Examples of a Creator include a person, an organization, or a service. Typically, the name of a Creator should be used to indicate the entity.
dc:identifier	An unambiguous reference to the resource within a given context. Recommended best practice is to identify the resource by means of a string conforming to a formal identification system.

dc:subject	The topic of the resource. Typically, the subject will be represented using key-words, key phrases, or classification codes. Recommended best practice is to use a controlled vocabulary.
dc:title	A name given to the resource.
dc:type	The nature or genre of the resource.
dcterms:bibliographicCitation	A bibliographic reference for the resource. Recommended practice is to include suffi-cient bibliographic detail to identify the resource as unambiguously as possible.
dcterms:issued	Date of formal issuance (e.g., publication) of the resource.
dcterms:partOf	A related resource in which the described resource is physically or logically includ-ed.
dcterms:references	A related resource that is referenced, cited, or otherwise pointed to by the described resource.
dcterms:tableOfContent	A list of subunits of the resource.
swrc:editor	Is used for defining the editors of the books.
swrc:journal	Is used for defining the title of the Journal
swrc:number	
swrc:pages	Is used for defining the number of pages for the Journal.
swrc:series	
swrc:volume	The name of an author or a title of a jour-nal or article in String format.
rdf:type	
rdfs:label	

rdfs:seeAlso	
owl:sameAs	An ontology which is the same as the one on the left side of the triplet.
foaf:homepage	The homepage of the author of a publication
foaf:maker	The creator of the article or publication
foaf:name	The name of the creator-author of a publication or article
foaf:page	The type or format of a recourse (e.c pdf, html, php, ps, or other).

Figure 3.3.6 Description of the properties in DBLP RDF format of data

Summarizing, the Figures 4.2.5 and 4.2.6 give a brief description of all the classes and the properties that can be used for querying DBLP resource by using SPARQL. The following examples make it easier to be understood.

```
PREFIX conf: <http://dblp.l3s.de/d2r/resource/conferences/>

SELECT  ?Name  ?WebPage
WHERE
{
?Author    foaf:name    ?Name .
?Paper     dc:creator    ?Author .
?Paper     dcterms:partOf    ?Conf .
?Conf      swrc:series    conf:pods .
}
```

LIMIT 4

The previous SPARQL query finds all the Authors' names who had a publication in *pods* series of Conferences, in String format. To achieve this result the query uses the properties foaf:name for the authors name, dc:creator for finding the person who is author of a publication. The property dcterms:partOf "says" that the publication is part of the conference which belong to *pods* series of conferences. Finally the retrieved answers are limited to four. So the results are: "Michael J. Maher", "Divesh Srivastava", "Jan Chomicki" and "Dina Q. Goldin".

In this example it is also defined a new prefix, the "conf" in order to succeed faster search and be easier to look straight to the pods series of conferences.

3.3.3 DBLP Endpoint

One of the main reasons that DBLP has been selected was that the particular database offers an endpoint which can be used for information retrieval with the help of SPARQL or SNORQL querying languages. The endpoint is part of the D2RQ Platform and can be used both online and locally by installing it on a PC.

The D2RQ[18] Platform is a system for accessing relational databases as virtual, read-only RDF graphs. It offers RDF-based access to the content of relational databases without having to replicate it into an RDF store. The use of D2RQ offers the ability to:

- query a non-RDF database using SPARQL
- access the content of the database as Linked Data over the Web

- create custom dumps of the database in RDF formats for loading into an RDF store
- access information in a non-RDF database using the Apache Jena API

The D2RQ Platform consists of:

- the *D2RQ Mapping Language*, a declarative mapping language for describing the relation between an ontology and an relational data model. More…
- the *D2RQ Engine*, a plug-in for the Jena Semantic Web toolkit, which uses the mappings to rewrite Jena API calls to SQL queries against the database and passes query results up to the higher layers of the frameworks.
- *D2R Server*, an HTTP server that provides a Linked Data view, a HTML view for debugging and a SPARQL Protocol endpoint over the database.

Finally, D2RQ is Open Source software and published under the Apache license. The most useful part of the D2RQ project for the specific dissertation was the D2R Server.

This one, is a tool for publishing the content of relational databases on the Semantic Web, a global information space consisting of Linked Data.

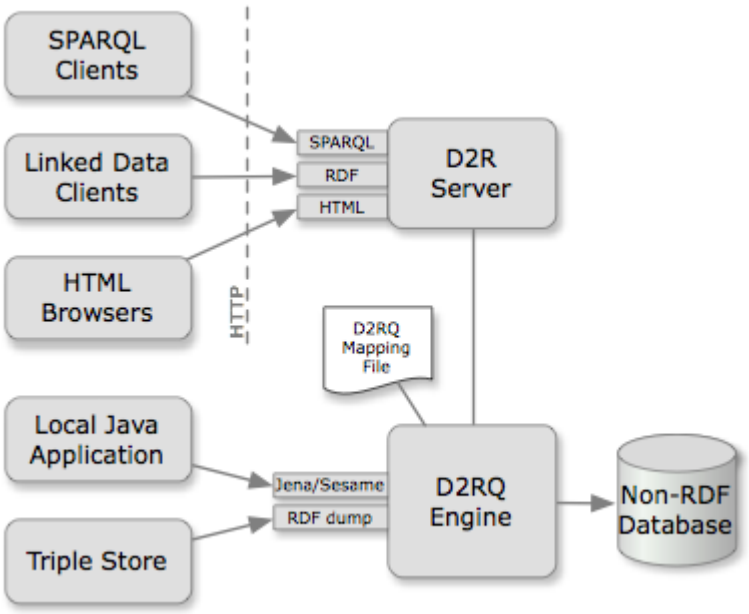


Figure 3.3.7 D2RQ presentation

Data on the Semantic Web is modelled and represented in RDF. D2R Server uses a customizable D2RQ mapping to map database content into this format, and allows the RDF data to be *browsed* and *searched* – the two main access paradigms to the Semantic Web.

Requests from the Web are rewritten into SQL queries via the mapping. This on-the-fly translation allows publishing of RDF from large live databases and eliminates the need for replicating the data into a dedicated RDF triple store.

Regarding the graphical interface it is a simple one and allows the user to write the whole query in Snorql. The server and its endpoint can be used from different organizations. The interface remains the same but with different prefixes for each one. So for the DBLP endpoint, the interface is following in the Figure 4.2.8 .

Figure 3.3.8 The graphical interface for SPARQL queries on d2r server

Regarding the Figure 4.2.8 the "Snorql: Exploring hhttp://dblp.l3s.de/d2r/sparql" is the address in which the user can access the DBLP. The following 10 rows under the "SPARQL" are the prefixes that are used for the rdf format of the data in dblp and are standard. The following space is for the user to add his queries and prefixes in order to retrieve the data that he wants. Finally, he has the option to retrieve the data in HTML format, XML, JSON and XML+XSLT.

3.4 QUERIES

This section presents the selected queries that have been chosen and the reasons why they have been chosen in order to find the best links and relationships between the profiles of the scientists and their publications in the semantic web.

3.4.1 Query 1

Giving the name of two scientists (i.e. the user's name and the other scientists name) a user can explore if he has any connection with another scientist, such as cooperation for the publication of some papers in the past. The following query shows all the titles of

the publications where both of the scientists are co-authors and the Conferences where they had been published.

```
SELECT  ?title ?Conference  WHERE{
    ?paper dc:creator  ?n.
    ?paper dc:creator  ?coauthor.
    ?paper dc:title  ?title.
  ?paper dcterms:partOf ?InCof.
 ?InCof dc:title ?Conference.
    FILTER (?n = "USER's Name" &&
                ?coauthor = "Name of a Scientist")
}
```

The SPARQL query above is using the properties dc:creator and dc:title and finds the articles where each of the two given names are authors and then returns these that are authored from both of them. For each one of the returned papers finds the Conferences where they were published. This is achieved with the help of the property dcterms:part Of.

3.4.2 Query 2

This one takes as input the names of two scientists and returns all the articles that are authored by both of them. It also returns the Journals where these articles where published.

```
SELECT  ?title ?Journal  WHERE{
    ?paper dc:creator  ?n.
    ?paper dc:creator  ?coauthor.
    ?paper dc:title  ?title.
  ?paper swrc:journal ?InJourn.
 ?InJourn dc:title ?Journal.
    FILTER (?n = "USER's Name" &&
                ?coauthor = "Name of a Scientist")
```

```
}
```

The query above works exactly in the same way as the “Query 1” but in contrast with that it is using the property swrc:journal instead of dcterms:part of , for finding the Journals that contain the published articles and not the Conferences.

3.4.3 Query 3

Giving the name of two scientists the user has the ability to find out the conferences where both of them had participate as the creators of the same or different published articles. It also returns the names of the conferences where they have been published. The following SPARQL query shows how exactly these results are calculated.

```
SELECT DISTINCT ?title ?pTitle1  ?pTitle2
WHERE {
 ?Author1    foaf:name    "User1" .
 ?Paper1    dc:creator    ?Author1 .
 ?Paper1    dcterms:partOf   ?InCof .
 ?Paper1    dc:title     ?pTitle1 .

 ?Author2    foaf:name    "User2" .
 ?Paper2    dc:creator    ?Author2 .
 ?Paper2    dcterms:partOf   ?InCof .
 ?Paper2    dc:title     ?pTitle2.

 ?InCof     swrc:series    ?Conference .
 ?InCof     dc:title   ?title .
 }
```

This one uses dc:creator property to find the articles that the given scientists have published and then by using the dcterms:partOf and swrc:series define that these articles

should be part of the same conferences. Finally the dc:title returns the names of the conferences and the titles of the articles that the two scientist had published in each of the conferences.

3.5 CONCLUSIONS

To summarize, the fourth chapter of this dissertation describes all the decisions that have been made in order the solution to be designed. At first it is presented the main problem of the semantic web and the reasons that is not fully adopted in the web. Then, in 4.1 it is presented all the available data resources and the reason that the final choice was the DBLP. After that, the DBLP data resource is analyzed with the s\description of its components, the classes and the properties that are used for the definition of the data in RDF format. One of the most important components of the fourth chapter is the 4.2 that describes the way that the SPARQL endpoint for the DBLP is working with the help of d2r server. Finally, in chapter 4.3 there are presented the best sparql queries for the mining of the results.

4 IMPLEMENTATION

The fifth chapter of this dissertation contains all the steps which were followed in order the tool to be created and the goal to be achieved. More specifically here are presented the technologies, the programming languages and the libraries that were used, the queries and the way that the results were ranked from these with the biggest relationship to these with no or small relationship for achieving the goal of the project, to find the scientific path between computer scientist that have semantic web's profile.

4.1 Back-end

For the creation of the tool it was finally selected Java as the programming language. The reasons which have also analysed in the Chapter 3.3 are the following:

- Easy to use.
- Object Oriented that gives more abilities to wright easier and reusable code.
- Provides features that can make the creation of a web application simple by providing local servers.
- Can add Apache Jena libraries for the usage of RDF data.

4.1.1 Java Tools

More specifically it was selected Java EE 6 with Glassfish Server 4.1 in order to create all the "logic" of the tool in the server side. This version of Enterprise Edition it is enhanced with EJB 3.1, Java Persistence API (JPA) 2.0, the new, yet already very successful, Java API for RESTful web services (JAX-RS), and the made-over JavaServer Faces (JSF) 2.0 specification. The enterprise Java platform has now matured to a degree where it can be both complete and lightweight.

Figure 4.1.1 Java Enterprise Edition

The figure above presents how the Java Enterprise Edition is connected with HTML5 and also meets the Enterprise Demands.

Regarding the Apache GlassFish Server 4.1[31], GlassFish is an open-source application server project started by Sun Microsystems for the Java EE platform and now sponsored by Oracle Corporation.GlassFish is the reference implementation of Java EE and as such supports Enterprise JavaBeans, JPA, JavaServer Faces, JMS, RMI, JavaServer Pages, servlets, etc. This allows developers to create enterprise applications that are portable and scalable, and that integrate with legacy technologies. Optional components can also be installed for additional services.

Both Java EE 6 and Apache GlassFish 4.1 are installed and used from NetBeans IDE 8.0.1. NetBeans IDE lets the user quickly and easily to develop Java desktop, mobile, and web applications, as well as HTML5 applications with HTML, JavaScript, and CSS. The IDE also provides a great set of tools for PHP and C/C++ developers. It is free and open source and has a large community of users and developers around the world.

To summarize, for the implementation of the server side of the tool it was used Net-beans IDE platform by adding the Apache Server 4.1 and the use of the external libra-ries that are provided from Apache Jena.

4.1.2 Server Side Analysis

Regarding the way that the code is organized. It is presented in the following Figure that there is one HTML file which is connected with a Bean Java file for the manipulation of

the server side features. The bean file it is also connected with another Java file the Qresult.java file.

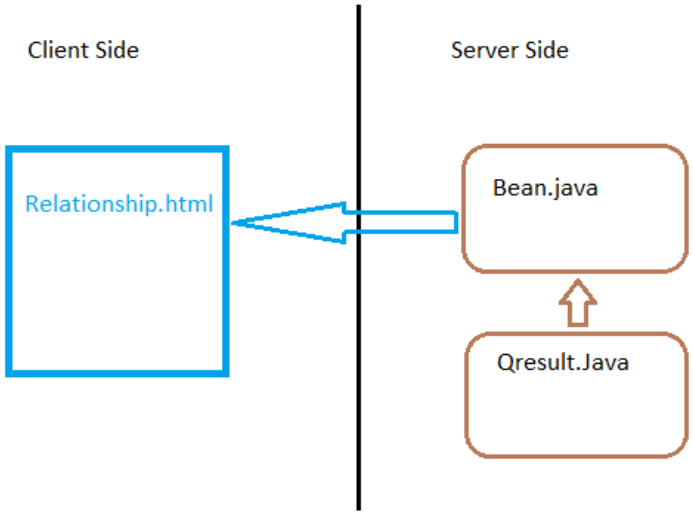


Figure 4.1.2 Presentation of the how the different files are connected

The most important file of the server side part of this project is the Bean.java file. To Begin with it contains several functions. The most important is the *"letscallsparql ()"* that contains all the predefined Parameterized queries which are used for retrieving data and were analyzed in the 4th chapter. As it is presented before the tool is connected to the Sparql Endpoint which is provided by the University of Trier[32] in the following link http://dblp.l3s.de/d2r/sparql by using the following command, *"QueryExecutionFactory.sparqlService("http://dblp.l3s.de/d2r/sparql", qs1.asQuery());"* which uses as first parameter the server's address and as second the predefined query. By this command a query is sent as a string to the endpoint and the data return in RDF format and can be converted in String format.

The aims of this projects is not only to find scientific relationships between two computer scientists but to rank these results. The ranking is achieved as following:

4. Closest connection between two scientists is that they have published together an article on a Conference or in a Journal. They are co-authors.
5. The second level of relationship between two scientists is that they had participated in the same conference or journal with their own article or paper.
6. The third level of relationship between the scientists is their background regarding the keywords of their semantic web.

The ranking idea has been implemented by an “if else” condition and it is a part of the information retrieval process. If the first condition returns results then they are presented to the user and the relationship is find, the two scientist are co-authors of a publication. If not then the second condition is checked and so on. In this point it should be mentioned that because of the absence of keywords in any scientist’s profile, it was not available any query which could have returned relationship in the background of them. So the solution was the mining of keywords from the titles of the publications. The process was the following. A simple Sparql Query returns the titles of all the publications of the two scientists. Then for each one, the titles are scanned for finding the most popular and common words. In the end both the common words in titles of the first and the second scientist are compared and if any similar keywords are found they are returned as the common background.

Finally another important function is the “onNodeSelect ()” . This function is used for the graphical representation of the results which will be analyzed in the subchapter 5.2 . For each one of the relationships between the two scientists it uses nodes that re connected with lines.

4.2 Front-end

It was reported before that the tool that was created during this dissertation project it was designed to be and used as a web application. The client-side format of the application should have been easy for usage and of course easy to be developed in a way that could communicate with the server and retrieve data fast.

4.2.1 Web Tools

Netbeans IDE 8.0.1 provides the ability to create web applications, as well as HTML5 applications with HTML, JavaScript, and CSS.

For the CSS it was used a standard css format from Themeforest[33] which provides a huge number of styles and themes.

In addition JSF and PrimeFaces[34], which is characterized as the ultimate UI framework for Java EE, gave the ability of using JavaScript and also made the connection with the server-side Java, simple.

4.2.2 Client Side Analysis

For the developing of the client-side part of the tool it was used HTML with JSF. To begin with, the application starts with the home page, the index.xhtml witch contains some information about the specific dissertation. The following Figure shows the homepage.

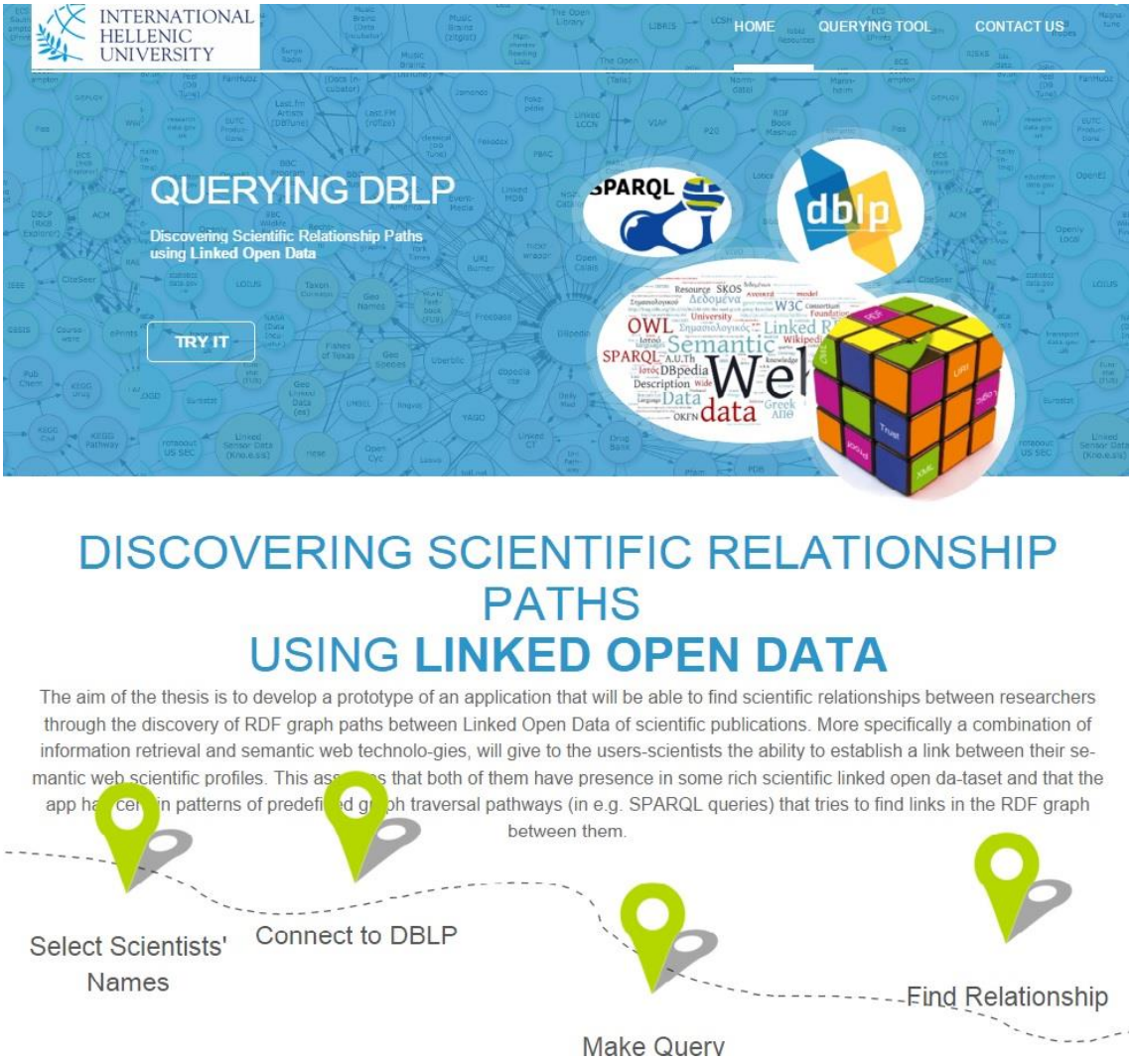


Figure 4.2.1 The homepage of the application

Regarding the tool there is one file, the relationship.html which contains all the code for the user interface. The first and only panel that it is not hidden when the application

starts is one input panel with id=inputPanel and contains two input texts where the user can give the names of two computer scientists in order to find how they are related. The following figure gives a picture of it.

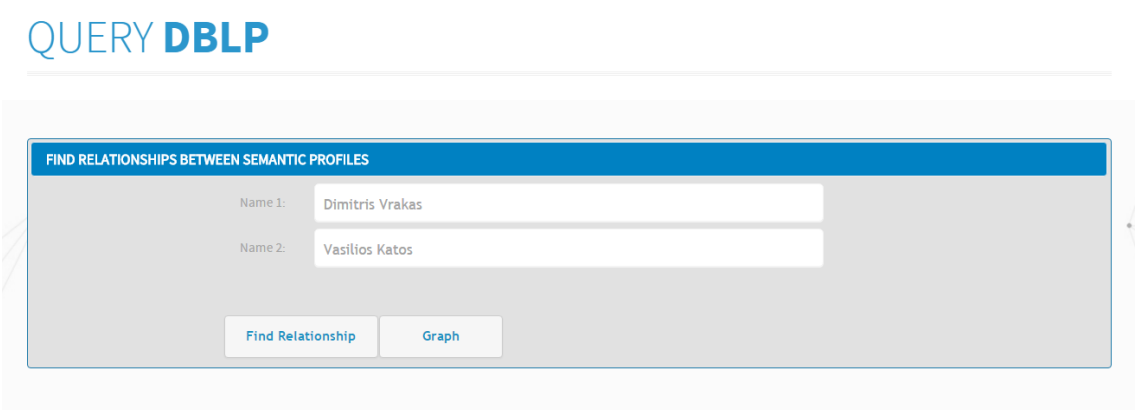


Figure 4.2.2 The input panel

After that there is a hidden output panel which contains the different tables for each one of the possible relationship between the scientists. Another one hidden output panel is the one that is used for the graphical representation of the data. More specifically the output panel in Figure 4.2.3 contains a table for the first relational level of the scientist which mean that they are co-authors of a published article in a conference or a Journal.

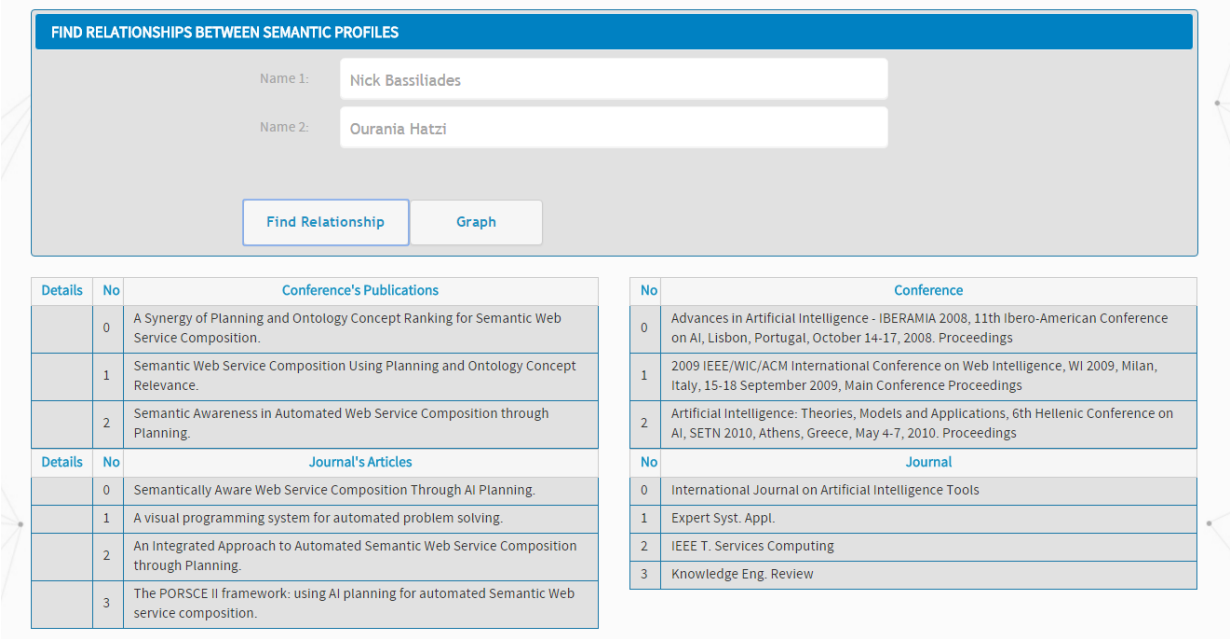


Figure 4.2.3 The output panel for the First relational level.

In this example the user gives two names as an input and by pushing the Find Relationship button the results that are returned are that scientists “Nick Bassiliades” and “Ourania Hatzi” are co-authors of three papers that had participated in three conferences, there can be seen both the titles of the papers and the titles of the conferences. The two scientists are also coauthors of four articles in Scientific Journals. Again the titles of the articles and the Journals are returned.

The Following Figure gives a graphical representation of the first relational level of the two scientists. Again the user gives the names of two scientists, in this example are again Nick Bassiliades and Ourania Hatzi.

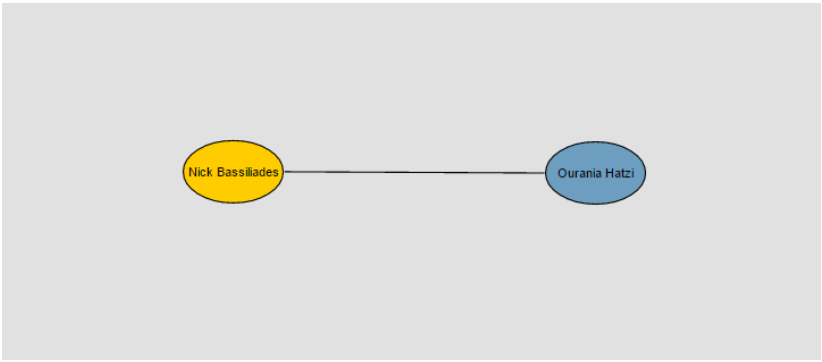


Figure 4.2.4 Graphical representation of the relationship

By pushing the “Graph” button the graphical output panel gives two bullets, each one has a name of a scientist and they are connected with a line. Then the user can click the one bullet and see the relationship. In the specific example as it was presented before the scientists are coauthors of three published papers in Conferences which here are presented with yellow color and four articles is Journals which are presented with blue color.

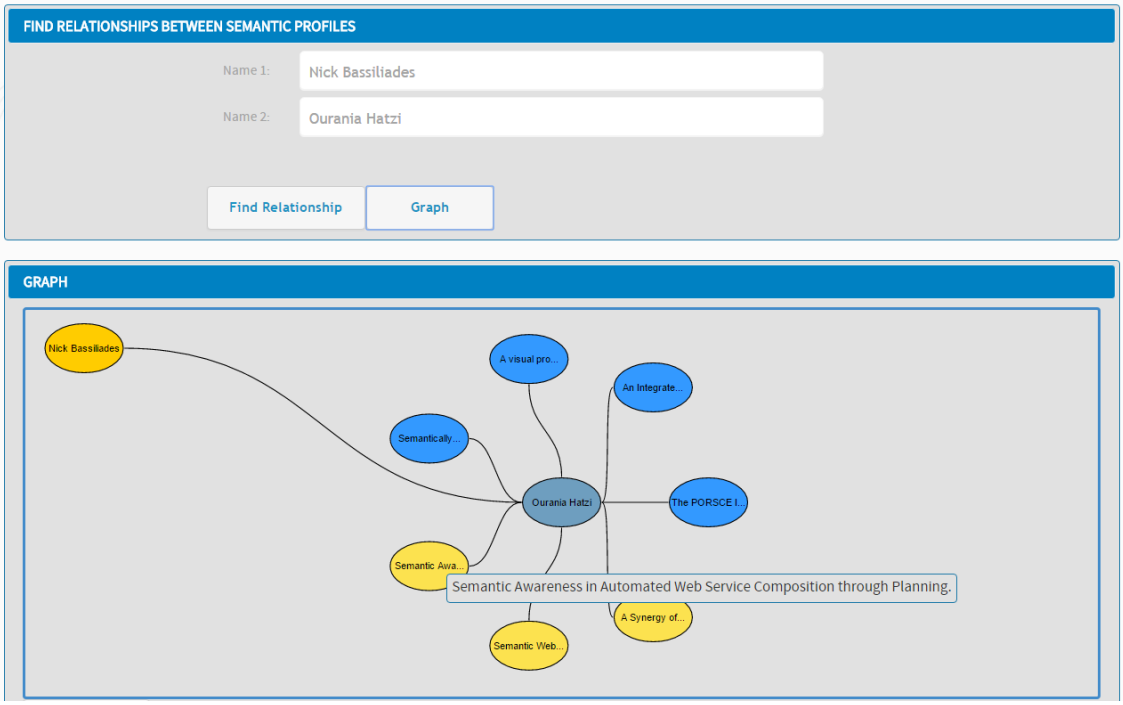


Figure 4.2.5 The graphical representation of the first relational level

If the two scientists are not co-authors of any publication then the second relational condition is checked. Now if any results are returned means that they have published their own article in the same Conference or Journal. The following example in Figure 4.2.5 takes as an input two names of computer scientists and returns the conferences that both of them had joined. In this example Patricia Melin and Allen Hobbs had published a paper in the same conference “Proceedings of the IEEE/IAFE 1995 Computational Intelligence for Financial Engineering, CIFEr 1995, New York City, USA, April 9-11, 1995”

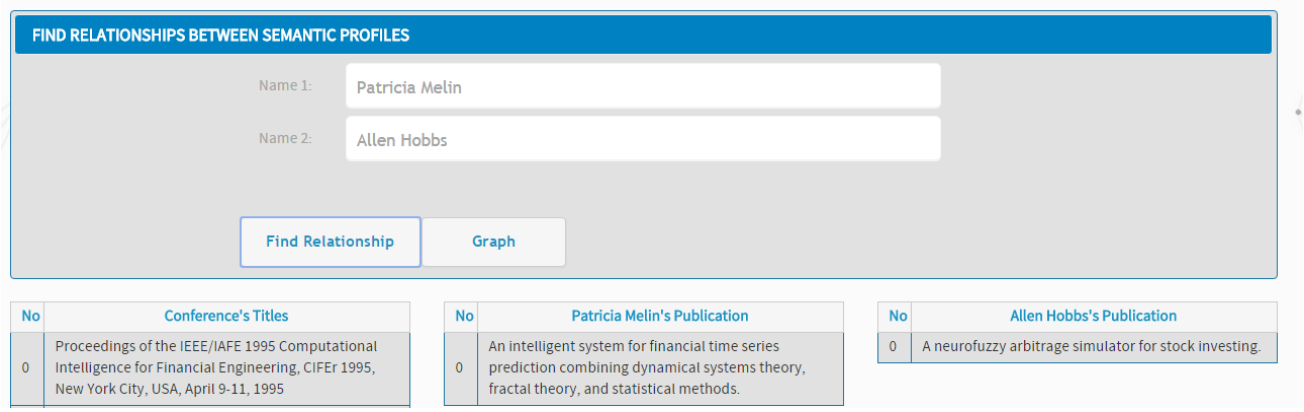


Figure 4.2.6 Second relational level. Joining the same conference

Again the user has the ability to see the graphical representation of the relationship between the two scientists by pushing the "Graph" button. The following Figure shows the graph.

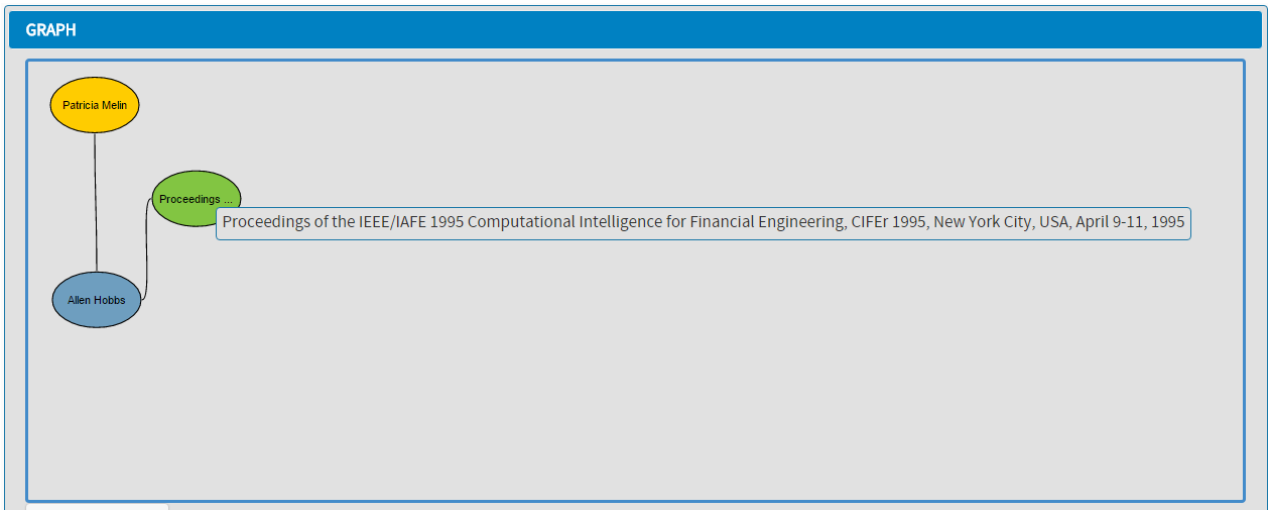
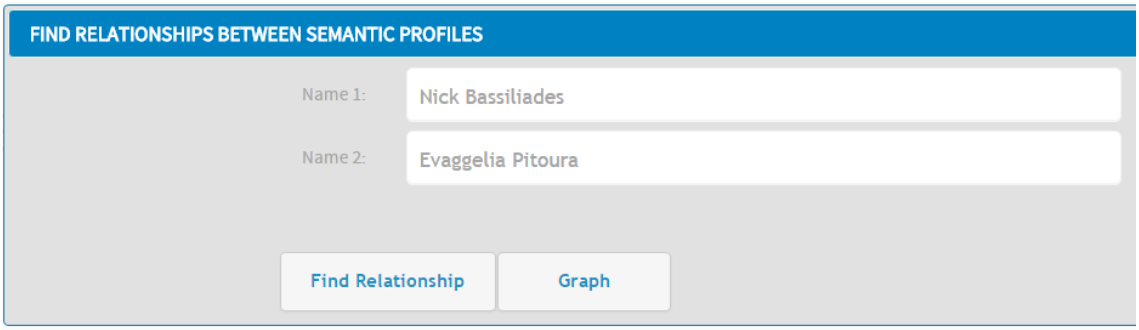


Figure 4.2.7 Graphical representation of the Relationship.

The yellow and blue bullet contain the names of the scientists while the green has the title of the conference that both of them had joined the same time.

Finally if the scientists are not connected with any publication, conference or journal then, it is checked their background by comparing the keywords that are used for the description of their publications. This is the third and final relational level of the tool. The Figure 4.2.7 shows this relationship.



No	Similar Background
0	Web
1	Data
2	Agents
3	Information
4	Services
5	Database

Figure 4.2.8 The third relational level, similarities in their scientific background

This example shows that the only relationship between the two scientists is their background and more specifically their publications are related with the “Web”, ”Data”, ”Agents”, “Database”, “Information”. This relationship can also be seen in graphical mode.

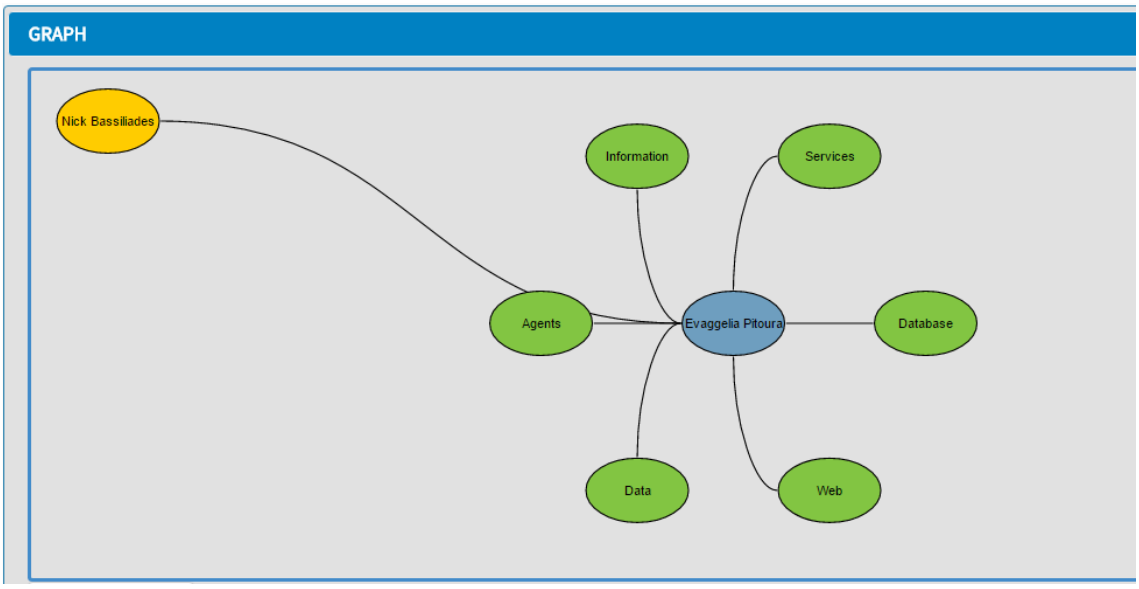


Figure 4.2.9 Graphical representation of the relationship based in the background.

This graph shows with the green bullet the keyword that both of the scientists uses for describing their publications and it is common between them.

4.3 Conclusion

The fifth chapter of this dissertation described the way that the tool was implemented. In the first subchapter were presented the tools and the technology that were used and also were presented in depth the way that the server side of the tool was created. The chapter 5.2 was dedicated to the front end of the tool. At first were presented all the available technologies and then the interface with screen shots and examples.

5 CONCLUSIONS

This chapter gives a general description of the dissertation. How it was implemented, which technologies were chosen and why, who were the goals and which of them had achieved. It also presented a critical view of what was created and what it would have been if it was created again from scratch. Finally it is presented the future work and the knowledge that was extracted throw the implementation of the project.

5.1 General Description

To begin with, the aim of the thesis was to develop a prototype of an application that will be able to find scientific relationships between researchers through the discovery of RDF graph paths between Linked Open Data of scientific publications. More specifically a combination of information retrieval and semantic web technologies, would give to the users-scientists the ability to establish a link between their semantic web scientific profiles. This assumes that both of them have presence in some rich scientific linked open dataset and that the app has certain patterns of predefined graph traversal pathways (in e.g. SPARQL queries) that tries to find links in the RDF graph between them. Some links could be “the two scientists have presented a paper at the same conference or one of them has cited the paper of the other or they have similar research interests”.

The first step was the research of the available technologies and scientific subjects that could have been able to make a contribution in the implementation process. This is presented in details in the second chapter, the “Literature review”. Firstly, it is referred the Semantic Web which provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is the basis of the project because the idea it is based on the ability that the Semantic Web offers, the reusability of Data throw applications. Then it is analyzed the RDF format of data which are the standards that are used for the presentation by the Semantic Web. The next subchapter is about the Linked Data and more specifically the Linked Open Data. To make the Web of Data a reality, it is important to have the huge amount of data on the Web

available in a standard format, reachable and manageable by Semantic Web tools. Furthermore, not only does the Semantic Web need access to data, but relationships among data should be made available, too, to create a Web of Data (as opposed to a sheer collection of datasets). This collection of interrelated datasets on the Web can also be referred to as Linked Data. In addition the second chapter contains analysis about the Querying languages that are used for the handling of the Linked Data. SPARQL is an RDF query language, that is, a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format. Finally the "Literature Review" contains also some information for the Information Retrieval process which is the idea where the dissertation was based. A user gives a query to the tool. Then results are extracted from the datasets and return to the user after they ranked.

After that follows the "Designing of the Solution". This one is the Fourth Chapter and describes the technologies and the reasons that they were chosen. More specifically starts with an investigation about all the available data resources that could be used. Finally it was selected the DBLP and the reason was that it follows all the following principles:

- Data in RDF format
- Data that conclude Computer Science
- Open/Free Data resources
- Endpoint for Semantic Querying Languages (e.g. SPARQL)

Then, all the features of DBLP are analyzed. Its components, the classes and the properties of the RDF format of the data and of course the Endpoint and the way that it works. The final subchapter contains all the SPARQL queries that was used in order to retrieve the data and find the relationships between the Computer Scientists. The fifth chapter describes how did the tool implemented. It is presented with details how did the server side part of the tool and which where the technologies that were used and how the connection to the DBLP endpoint had achieved. Then it also presented the client part Of the tool followed from the technologies that were used and of course followed from images that present the different usage scenarios. There are used examples for findng all the

possible relationships between two computer scientists. The relation levels are the following:

- The scientists are coauthors of a publication
- The scientists had joined the same Conferences with their own publications
- The scientists have similar scientific background

Finally, follows the comparison of the available programming languages for choosing the one that can be used for the implementation of the tool.

5.2 Solving the Problems

The chapter 4.1 addresses all the problems that should be solved and studied in order the dissertation to be finished and the gives the best results, It also a way to understand the success of this thesis in the sector of Semantic Web technologies. This subchapter, presents the achieved solutions of the objective problems and the problems that where part of the development of the tool.

Regarding the Data Resources that contain data in semantic webs' format, as it is presented in the chapter 3.2 were found a good number of resources with interesting information. Finally it was selected the DBLP because it was totally fit with the goal and the tools that were available for the development of the application.

The challenge for Linked Open Data to be found, again it was faced by the DBLP. This data resource had all the data Open for internal usage by the offered sparql endpoint or external usage by free downloading of the data in RDF format.

The next challenge was the study of how this data can be linked. What is the idea of the Linked Data and the understanding of the way that they can be reclaimed? The Open Linked Data are graphs that can be connected with semantically meaningful paths. To achieve this connections between the semantic data it is used the RDF format of data which specifies the type of data and makes it easy to be connected.

Despite the good number of different Linked Data resources that were found, it was not feasible in this dissertation the data to be connected or Linked. The reason is that the only data resource that it was free accessible and had information respecting the computer science it was the DBLP. As a result the tool it could not be implemented in a way

that could connect DBLP data with Data from another Data Resource and retrieve some information that are not concluded in it. Eventually the goal that was set in the beginning to Link data from different data resources didn't achieved.

Regarding the problems that focus on the building and the designing of the tool. The process of finding semantical scientific paths between two scientists was achieved. The Querying language Sparql gave the ability via the appropriate queries to retrieve the right information for finding these relations. More specifically the tow first relational levels where created successfully, by finding if the scientists are coauthors of some publications or had joined the same conferences. Regarding the third relational level, it was not able to be achieved through any query. The reason is that DBLP does not contain keywords for the background of the scientist. As a result it had to be found another way to retrieve these information. This was achieved by finding the most important-keywords word of the publications titles and use them for finding similarities.

The final problem was an addition usage of the tool. The ability to use a query that returns scientists that have published papers with a topic identical or similar (semantically similar, using an ontology of topics) to the topic in question. This was not able to be solved for two reasons. First of all, as it is pointed before the DBLP does not has any information about keywords. The second reason was that there is not any Open Data source with similar information and the tool was not able to Link Data between different data resources in order to retrieve the best results. So this problem was not solved.

5.3 Gaining Knowledge

This dissertation required the study of several computer science sectors. These studies gave some interesting information about the Semantic Web, the Open Linked Data, and the Information Retrieval Process.

The first and most important results that gave the important information about the prefaced computer science sectors are that the main subject of the dissertation was successfully captured in a web application. Open Linked data which contain semantic profiles

of computer scientists gave through this tool the ability for everyone to find scientific relationships between them.

Through this process it was made comprehensible the meaning of the Semantic Web,that it is a framework which allows data to be shared and reused across applications. In addition it was studied in depth the format that these data have and what does it mean the representation of them, how the triplets work and why it is used the RDF format under the 3WC. The fact that the data were retrieved with the help of a querying Language such as SPARQL gave the ability the queries and the language to be studied in depth in order to succeed the best results.

The research for finding the available semantic Data Resources gave important information about the number of the communities that are interesting and focus on web where all the data will be Linked and the whole web would be converted in semantic web with data that will be accessible from applications, enterprises and meet the expectations of Tim Burnes-Lee for a web of data that can be accessed from machines.

A good knowledge that was gained after this dissertation, it was the importance of the data to be free or not, for everyone. Theoretically it was studied the difference between Linked Data and Linked Open Data with the help of the 5 star schema in the chapter 2.4.2 . In practice the differences were much bigger. This was the reason that the web application that was created in this dissertation did not managed to cover all the dissertation subjects' challenges. The lack of Open Data Resources did not allow the usage of potential keywords in order to find the third relational level by connecting different Data Recourses which is the real meaning of the Linked Data.

5.4 Future Work

This subchapter describes all the potential changes that could be done in the specific tool that was created for this dissertation and the future work that could be done in order the tool to be more functional and compatible for more tech devices.

To begin with, the dissertation goals for creating an application that could be able to find scientific relationship paths between the semantic profiles of computer scientists was succeeded. Although there are some functionalities that could be redesigned in the future in order to achieve better results. More specifically:

- The third relational level-common scientific background-was not able to be found by querying DBLP but by extracting keywords from the titles of the publications and finding similar keywords. This one can be redesigned in the future and be implemented a new solution be comparing the scientists background from their semantic profiles in other Data Resources which will contain keywords in RDF format.
- Another one goal that was not achieved in this dissertation is the ability to connect different data resources and see the real power of Linked Data. As it was mentioned in the chapter 5.2 the reason that it was not achieved it was the lack of Linked Open Data resources. So as a future work it would be the effort to find more of these resources and find also a way to connect them in order to be highlighted the effectiveness of the application.
- Then, in the future the application could be able to give to the users the abilities not only to find relational paths between two scientists but to have the fluency to make searches regarding a conferences title or the more advanced users to be able to create their own SPARQL queries and retrieve the information that they like.
- Finally this application was designed as a web application. In the future it would be great to have also a mobile applications version that could be used by android or IOS smart devices and make it easy accessible.

Bibliography

[1] https://en.wikipedia.org/wiki/Information_retrieval#cite_note-goodron2000-1

[2] Mooers, C. N. (1951). Zatocoding applied to mechanical organization of knowledge. American Documentation, 2, 20-32.

[3] Frakes, William B. (1992). Information Retrieval Data Structures & Algorithms. Prentice-Hall, Inc. ISBN 0-13-463837-9.

[4] Ricardo Baeza-Yates, Berthier Ribeiro-Neto (2011) Modern Information Retrieval: The Concepts and Technology behind Search (2nd Edition) (ACM Press Books)

[5]http://www.w3.org/standards/semanticweb/data

[6] http://www.w3.org/DesignIssues/LinkedData

[7] http://linkeddata.org/

[8] Berners-Lee, Tim; James Hendler; Ora Lassila (May 17, 2001). "The Semantic Web". Scientific American Magazine.

[9] Optimized Index Structures for Querying RDF from the Web Andreas Harth, Stefan Decker, 3rd Latin American Web Congress, Buenos Aires, Argentina, October 31 to November 2, 2005

[10] http://www.w3.org/TR/PR-rdf-syntax/

[11] Jim Rapoza (2 May 2006). "SPARQL Will Make the Web Shine". eWeek.

[12] Segaran, Toby; Evans, Colin; Taylor, Jamie (2009). Programming the Semantic Web. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol

[13] http://jena.apache.org/

[14] (http://richard.cyganiak.de/)

[15] (http://richard.cyganiak.de/)

[16] http://authors.acm.org/

[17] Kulkarni, A. V.; Aziz, B.; Shams, I.; Busse, J. W. (2009). "Comparisons of Citations in Web of Science, Scopus, and Google Scholar for Articles Published in General Medical Journals"

[18] http://d2rq.org/d2r-server

[19] W3C Semantic Web Activity. Marja-Riitta Koivunen and Eric Miller.

[20] Linked-Data The Story So Far. Christian Bizer, Freie Universität Berlin, Germany.

Tom Heath, Talis Information Ltd, United Kingdom.Tim Berners-Lee, Massachusetts Institute of Technology, US.

[21] Querying Semantic Web Data With SPARQL. Marcelo Arenas, Department of Computer Science PUC Chile. Jorge Perez, Department of Computer Science Universidad de Chile.

[22] ACM Digital Library. http://dl.acm.org/

[23] Scopus. http://www.scopus.com/

[24] DBLP. http://dblp.uni-trier.de/

[25] arXiv. http://arxiv.org/

[26] Springer. http://www.springer.com/gp/

[27] Semantic Web Dog Food.

[28] https://jena.apache.org/

[29] http://www.dotnetrdf.org/

[30] https://rdfsharp.codeplex.com/

[31] https://glassfish.java.net/

[32] https://www.uni-trier.de/index.php?id=48&L=2

[33] http://themeforest.net/?ref=iclicksol

[34] http://www.primefaces.org/[35] https://en.wikipedia.org/wiki/SPARQL

[36] https://en.wikipedia.org/wiki/FOAF_%28ontology%29

Appendix

Application's Back-End Code

Bean.java

```
@ManagedBean
@ViewScoped
public class Bean implements Serializable {

   private MindmapNode root;
   private MindmapNode co;
   List<String> nodelistconf = new ArrayList<String>();
   List<String> nodelistjrn = new ArrayList<String>();
   List<String> nodelistconferences= new ArrayList<String>();
   List<String> nodelistjournals = new ArrayList<String>();
   List<String> nodelistbackgr = new ArrayList<String>();
   List<String> titles1 = new ArrayList<String>();
   List<String> titles2 = new ArrayList<String>();

   ////////////////////////////////
   private String name;

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
      root = new DefaultMindmapNode(name, "Name 1", "FFCC00", false);//
   }

   private String coauthor;

   public String getCoauthor() {
      return coauthor;
   }

   public void setCoauthor(String coauthor) {
      this.coauthor = coauthor;
      co = new DefaultMindmapNode(coauthor, "Name 2", "6e9ebf", true);//
      root.addNode(co);
   }
```

```
ResultSet results;
ResultSet resultsnew;
String see;
String see1;

public String getSee() {
   return see;
}

public void setSee(String see) {
   this.see = see;
}

public  List<Qresult> qresults= new ArrayList<Qresult>();
public  List<Qresult> qresultsConf= new ArrayList<Qresult>();
public  List<Qresult> qresultsAuth= new ArrayList<Qresult>();
public  List<Qresult> qresultsArt= new ArrayList<Qresult>();
public  List<Qresult> qresultsArtJourn= new ArrayList<Qresult>();
public  List<Qresult> qresultsConfnew= new ArrayList<Qresult>();
public  List<Qresult> qresultstitleArt= new ArrayList<Qresult>();
public  List<Qresult> qresultstitle1Art= new ArrayList<Qresult>();
public  List<Qresult> qresultstitle2Art= new ArrayList<Qresult>();
public  List<Qresult>  qresultsback = new ArrayList<Qresult>();


public void setQresultsback(List<Qresult> qresultsback){
   this.qresultsback=qresultsback;
}

public List<Qresult> getQresultsback(){
   return qresultsback;
}

public void setQresultstileArt(List<Qresult> qresultstitleArt) {
        this.qresultstitleArt = qresultstitleArt;
   }

public List<Qresult> getQresultstitleArt() {
   return qresultstitleArt;
}

public void setQresultstile2Art(List<Qresult> qresultstitle2Art) {
        this.qresultstitle2Art = qresultstitle2Art;
   }
```

```
public List<Qresult> getQresultstitle2Art() {
   return qresultstitle2Art;
}

public void setQresultstile1Art(List<Qresult> qresultstitle1Art) {
        this.qresultstitle1Art = qresultstitle1Art;
   }

public List<Qresult> getQresultstitle1Art() {
   return qresultstitle1Art;
}

public void setQresultsArtJourn(List<Qresult> qresultsArtJourn) {
        this.qresultsArtJourn = qresultsArtJourn;
   }

public List<Qresult> getQresultsArtJourn() {
   return qresultsArtJourn;
}

public void setQresultsArt(List<Qresult> qresultsArt) {
        this.qresultsArt = qresultsArt;
   }

public List<Qresult> getQresultsArt() {
   return qresultsArt;
}


public void setQresultsAuth(List<Qresult> qresultsAuth) {
        this.qresultsAuth = qresultsAuth;
   }

public List<Qresult> getQresultsAuth() {
   return qresultsAuth;
}

public void setQresults(List<Qresult> qresults) {
        this.qresults = qresults;
   }

public List<Qresult> getQresults() {
   return qresults;
}

public void setQresultsConf(List<Qresult> qresultsConf) {
```

```
        this.qresultsConf = qresultsConf;
    }

public List<Qresult> getQresultsConf() {
    return qresultsConf;
}

public static Qresult qresult = new Qresult();

public void setQresult(Qresult qresult){
    Bean.qresult=qresult;
}

public Qresult getQresult(){
    return qresult;
}

public static Qresult qresultConf = new Qresult();

public void setQresultConf(Qresult qresultConf){
    Bean.qresultConf=qresultConf;
}

public Qresult getQresultConf(){
    return qresultConf;
}

public static Qresult qresultAuth = new Qresult();

public void setQresultAuth(Qresult qresultAuth){
    Bean.qresultAuth=qresultAuth;
}

public Qresult getQresultAuth(){
    return qresultAuth;
}

public void setQresultsConfnew(List<Qresult> qresultsConfnew) {
        this.qresultsConfnew = qresultsConfnew;
    }

public List<Qresult> getQresultsConfnew() {
    return qresultsConfnew;
}

//////////////////////////////////////////////////////////
```

```
boolean render1=false;
boolean render2=false;
boolean render3=false;
boolean render4=true;
boolean render5=false;
boolean render6=false;

public boolean panelun(){
   return render4;
}

public boolean panel1(){
   return render1;
}

public boolean panel2(){
   return render2;
}

public boolean panel3(){
   return render5;
}

public boolean panel4(){
   return render6;
}

public boolean graph(){
  return render3;
}

public void graphbutton(){
   render3=true;
   render4=false;
}

public void closebutton(){
   render3=false;
   render4=true;
}

//////////////////////////////////////////////////////////

public void letscallSparql(){
```

```
/******Re-Intit***************/
this.reinit();



/*****************************/


//Qresult qresult = new Qresult();
//List<Qresult> qresults = new ArrayList<Qresult>();
/////////////////Simple working Query for DBLP//////////////////////////////////////////////////////
//String author="Nick Bassiliades";
//String coauthor="Ourania Hatzi";


/************************************ QUERY 1 ***********************************************/
////////////Finding the coauthors and the Conferences/Journals where they published the Paper//////////////
ParameterizedSparqlString qs1 = new ParameterizedSparqlString(
    "prefix d2r: <http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2r-server/config.rdf#>\n"+
    "prefix swrc: <http://swrc.ontoware.org/ontology#>\n"+
    "prefix dcterms: <http://purl.org/dc/terms/>\n"+
    "prefix xsd: <http://www.w3.org/2001/XMLSchema#>\n"+
    "prefix dc: <http://purl.org/dc/elements/1.1/>\n"+
    "prefix map: <file:///home/diederich/d2r-server-0.3.2/dblp-mapping.n3#>\n"+
    "prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n"+
    "prefix foaf: <http://xmlns.com/foaf/0.1/>\n"+
    "prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>\n"+
    "prefix owl: <http://www.w3.org/2002/07/owl#>\n"+
    "\n" +
    "SELECT DISTINCT ?title ?Conference WHERE{\n"+
    "?paper dc:creator ?n.\n"+
    "?paper dc:creator ?coauthor.\n"+
    "?paper dc:title ?title.\n"+
    "?paper dcterms:partOf ?InCof.\n"+
    "?InCof dc:title ?Conference.\n"+
    "FILTER (?n = \""+name+"\" &&\n"+
    "?coauthor = \""+coauthor+"\")\n"+
    "}\n"+
    "LIMIT 5");

ParameterizedSparqlString qs11 = new ParameterizedSparqlString(
    "prefix d2r: <http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2r-server/config.rdf#>\n"+
    "prefix swrc: <http://swrc.ontoware.org/ontology#>\n"+
```

```
"prefix dcterms: <http://purl.org/dc/terms/>\n"+
"prefix xsd: <http://www.w3.org/2001/XMLSchema#>\n"+
"prefix dc: <http://purl.org/dc/elements/1.1/>\n"+
"prefix map: <file:///home/diederich/d2r-server-0.3.2/dblp-mapping.n3#>\n"+
"prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n"+
"prefix foaf: <http://xmlns.com/foaf/0.1/>\n"+
"prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>\n"+
"prefix owl: <http://www.w3.org/2002/07/owl#>\n"+
"\n" +
"SELECT DISTINCT ?title ?Journal WHERE{\n"+
"?paper dc:creator ?n.\n"+
"?paper dc:creator ?coauthor.\n"+
"?paper dc:title ?title.\n"+
"?paper swrc:journal ?InJour.\n"+
"?InJour dc:title ?Journal.\n"+
"FILTER (?n = \""+name+"\" &&\n"+
"?coauthor = \""+coauthor+"\")\n"+
"}\n"+
"LIMIT 5");
```

```
/************************************ QUERY 2 ***********************************************/
////////////// Returns the Conferences where the two Scientists had participated ///////////////

ParameterizedSparqlString qs2 = new ParameterizedSparqlString(
"prefix d2r: <http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2r-server/config.rdf#>\n"+
"prefix swrc: <http://swrc.ontoware.org/ontology#>\n"+
"prefix dcterms: <http://purl.org/dc/terms/>\n"+
"prefix xsd: <http://www.w3.org/2001/XMLSchema#>\n"+
"prefix dc: <http://purl.org/dc/elements/1.1/>\n"+
"prefix map: <file:///home/diederich/d2r-server-0.3.2/dblp-mapping.n3#>\n"+
"prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n"+
"prefix foaf: <http://xmlns.com/foaf/0.1/>\n"+
"prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>\n"+
"prefix owl: <http://www.w3.org/2002/07/owl#>\n"+
"\n" +
"SELECT DISTINCT ?title ?pTitle1 ?pTitle2 WHERE {\n"+
 "?Author     foaf:name      ?nn.\n"+
 "?Paper1     dc:creator     ?Author.\n"+
 "?Paper1     dcterms:partOf   ?InCof.\n"+
 "?Paper1     dc:title       ?pTitle1.\n"+
 "?Author2    foaf:name      ?n.\n"+
 "?Paper2     dc:creator     ?Author2.\n"+
 "?Paper2     dcterms:partOf   ?InCof.\n"+
```

```
        "?Paper2     dc:title      ?pTitle2.\n"+
        "?InCof     swrc:series      ?Conference.\n"+
        "?InCof dc:title ?title.\n"+
        "FILTER (?nn = \""+name+"\" &&\n"+
        "?n = \""+coauthor+"\")\n"+
        "}\n"+
        "LIMIT 5");

    ParameterizedSparqlString qs22 = new ParameterizedSparqlString(
        "prefix d2r: <http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2r-server/config.rdf#>\n"+
        "prefix swrc: <http://swrc.ontoware.org/ontology#>\n"+
        "prefix dcterms: <http://purl.org/dc/terms/>\n"+
        "prefix xsd: <http://www.w3.org/2001/XMLSchema#>\n"+
        "prefix dc: <http://purl.org/dc/elements/1.1/>\n"+
        "prefix map: <file:///home/diederich/d2r-server-0.3.2/dblp-mapping.n3#>\n"+
        "prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n"+
        "prefix foaf: <http://xmlns.com/foaf/0.1/>\n"+
        "prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>\n"+
        "prefix owl: <http://www.w3.org/2002/07/owl#>\n"+
        "\n" +
        "SELECT DISTINCT ?title ?pTitle1 ?pTitle2 WHERE {\n"+
        "?Author     foaf:name      ?nn.\n"+
        "?Paper1     dc:creator     ?Author.\n"+
        "?Paper1     swrc:journal   ?InCof.\n"+
        "?Paper1     dc:title       ?pTitle1.\n"+
        "?Author2    foaf:name      ?n.\n"+
        "?Paper2     dc:creator     ?Author2.\n"+
        "?Paper2     swrc:journal   ?InCof.\n"+
        "?Paper2     dc:title       ?pTitle2.\n"+
        "?InCof dc:title ?title.\n"+
        "FILTER (?nn = \""+name+"\" &&\n"+
        "?n = \""+coauthor+"\")\n"+
        "}\n"+
        "LIMIT 5");

/*******************************************************************************
***************/
    /************************************ QUERY 3
*********************************************/
    ////////////// Returns the titles of articles for a scientist ///////////////
    ParameterizedSparqlString qs3 = new ParameterizedSparqlString(
        "prefix d2r: <http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2r-server/config.rdf#>\n"+
        "prefix swrc: <http://swrc.ontoware.org/ontology#>\n"+
        "prefix dcterms: <http://purl.org/dc/terms/>\n"+
        "prefix xsd: <http://www.w3.org/2001/XMLSchema#>\n"+
        "prefix dc: <http://purl.org/dc/elements/1.1/>\n"+
```

```
        "prefix map: <file:///home/diederich/d2r-server-0.3.2/dblp-mapping.n3#>\n"+
        "prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n"+
        "prefix foaf: <http://xmlns.com/foaf/0.1/>\n"+
        "prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>\n"+
        "prefix owl: <http://www.w3.org/2002/07/owl#>\n"+
        "\n" +
        "SELECT DISTINCT ?title WHERE {\n"+
         "?paper     dc:creator     ?n.\n"+
         "?paper     dc:title      ?title.\n"+
         "FILTER (?n = \""+name+"\")}");

/*******************************************************************************
****************/
    ParameterizedSparqlString qs33 = new ParameterizedSparqlString(
        "prefix d2r: <http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2r-server/config.rdf#>\n"+
        "prefix swrc: <http://swrc.ontoware.org/ontology#>\n"+
        "prefix dcterms: <http://purl.org/dc/terms/>\n"+
        "prefix xsd: <http://www.w3.org/2001/XMLSchema#>\n"+
        "prefix dc: <http://purl.org/dc/elements/1.1/>\n"+
        "prefix map: <file:///home/diederich/d2r-server-0.3.2/dblp-mapping.n3#>\n"+
        "prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n"+
        "prefix foaf: <http://xmlns.com/foaf/0.1/>\n"+
        "prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>\n"+
        "prefix owl: <http://www.w3.org/2002/07/owl#>\n"+
        "\n" +
        "SELECT DISTINCT ?title WHERE {\n"+
         "?paper     dc:creator     ?n.\n"+
         "?paper     dc:title      ?title.\n"+
         "FILTER (?n = \""+coauthor+"\")}");



//org.apache.jena.rdf.model.Literal london = ResourceFactory.createLangLiteral( "London", "en" );
    //qs.setParam("label", (RDFNode) london);

    //System.out.println( qs );

    /***********************************Query                                    Checking
************************************************/

    QueryExecution exec;
    exec = QueryExecutionFactory.sparqlService( "http://dblp.l3s.de/d2r/sparql", qs1.asQuery() );

    results = ResultSetFactory.copyResults( exec.execSelect() );

    int flag=0;
```

```
System.out.println("Checking coauthors");
if(results.hasNext()){          //First Query for finding Co-Authors

  while ( results.hasNext() ) {
    see1 = results.next().get( "title" ).toString();
    nodelistconf.add(see1);////////////Conference Papars nodes
    qresult.rowstring=see1;
    qresults.add(qresult);
    qresult = new Qresult();
    coauthor(see1);
  }
  results = ResultSetFactory.copyResults( exec.execSelect() );
  while ( results.hasNext() ) {
    see = results.next().get( "Conference" ).toString();
    qresult.rowstringnew=see;
    qresultsConf.add(qresult);
    //System.out.println(see);

    qresult = new Qresult();
  }

  /////////////////Sub-query for Co-Authors in Journal Articles////////////////////////////////
  exec = QueryExecutionFactory.sparqlService( "http://dblp.l3s.de/d2r/sparql", qs11.asQuery() );
  results = ResultSetFactory.copyResults( exec.execSelect() );
  while ( results.hasNext() ) {
    see = results.next().get( "title" ).toString();
    nodelistjrn.add(see);////////////Journal Articles nodes
    qresult.rowstringArt=see;
    qresultsArt.add(qresult);
    System.out.println(see);
    qresult = new Qresult();
  }
  results = ResultSetFactory.copyResults( exec.execSelect() );
  while ( results.hasNext() ) {
    see1 = results.next().get( "Journal" ).toString();
    qresult.rowstringArtJourn=see1;
    qresultsArtJourn.add(qresult);
    //System.out.println(see);
    qresult = new Qresult();
  }
  render1=true;
  flag=1;
}
else if(flag==0){          //Second Query for finding the Conferences that both of the scientists had
Joined
  System.out.println("No papers, lets check CONFERENCES");
```

```
exec = QueryExecutionFactory.sparqlService( "http://dblp.l3s.de/d2r/sparql", qs2.asQuery() );
results = ResultSetFactory.copyResults( exec.execSelect() );
if(results.hasNext()){
  //System.out.println("LLLLLLLLLLLLLLLLLLLLLLLLLLLLA");
  while ( results.hasNext() ) {
    see = results.next().get( "title" ).toString();
    nodelistconferences.add(see);
    qresult.rowstring=see;
    qresults.add(qresult);
    System.out.println(see);
    qresult = new Qresult();
  }
  results = ResultSetFactory.copyResults( exec.execSelect() );
  while ( results.hasNext() ) {
    see = results.next().get( "pTitle1" ).toString();
    qresult.rowstringnew=see;
    qresultsConf.add(qresult);
    //System.out.println(see);

    qresult = new Qresult();
  }
  results = ResultSetFactory.copyResults( exec.execSelect() );
  while ( results.hasNext() ) {
    see = results.next().get( "pTitle2" ).toString();
    qresult.rowstringnew2=see;
    qresultsConfnew.add(qresult);
    //System.out.println(see);

    qresult = new Qresult();
  }
  render2=true;
  flag=1;
}
////////////Sub-Query for Articles//////////////////////////
/*System.out.println("No CONFERENCES, Lets check for JOURNALS");
exec = QueryExecutionFactory.sparqlService( "http://dblp.l3s.de/d2r/sparql", qs22.asQuery() );
results = ResultSetFactory.copyResults( exec.execSelect() );
if(results.hasNext()){
  while ( results.hasNext() ) {
    see = results.next().get( "title" ).toString();
    nodelistjournals.add(see);
    qresult.rowstring=see;
    qresultstitleArt.add(qresult);
    System.out.println(see);
    qresult = new Qresult();
  }
```

```
        results = ResultSetFactory.copyResults( exec.execSelect() );
        while ( results.hasNext() ) {
          see = results.next().get( "pTitle1" ).toString();
          qresult.rowstringnew=see;
          qresultstitle1Art.add(qresult);
          //System.out.println(see);

          qresult = new Qresult();
        }
        results = ResultSetFactory.copyResults( exec.execSelect() );
        while ( results.hasNext() ) {
          see = results.next().get( "pTitle2" ).toString();
          qresult.rowstringnew2=see;
          qresultstitle2Art.add(qresult);
          //System.out.println(see);

          qresult = new Qresult();
        }
        render2=true;
        flag=1;
      }
      /*else{
        flag=1;
      }*/
    }
    if(flag==0){ ////////////////Query for finding background
      System.out.println("Checking Background");
      exec = QueryExecutionFactory.sparqlService( "http://dblp.l3s.de/d2r/sparql", qs3.asQuery() );
      results = ResultSetFactory.copyResults( exec.execSelect() );
      List<String> commons = new ArrayList<String>();
      List<String> commons2 = new ArrayList<String>();

      if(results.hasNext()){
        while ( results.hasNext() ) {
          see = results.next().get( "title" ).toString();
          titles1.add(see);
          qresult.rowstring=see;
          qresultstitleArt.add(qresult);
          //System.out.println(see);
          qresult = new Qresult();
        }


       // for(int j=0;j<10;j++){
          Map<String,Integer> words_count = new HashMap<String,Integer>();
          Integer frequency = null;
```

```
String mostFrequent = null;
for(String st:titles1){
   String[] words = st.split("\\s+");
   for(int i=0;i<words.length;i++){
       String s = words[i];
       if(words_count.keySet().contains(s))
       {
           Integer count = words_count.get(s) + 1;
           words_count.put(s, count);
       }
       else{
           words_count.put(s, 1);
       }
   }

   for(int i=0;i<words.length;i++){
       List<String> slist = new ArrayList<String>();
       slist.add("A");
       slist.add("a");
       slist.add("An");
       slist.add("an");
       slist.add("as");
       slist.add("at");
       slist.add("and");
       slist.add("or");
       slist.add("of");
       slist.add("Of");
       slist.add("on");
       slist.add("On");
       slist.add("in");
       slist.add("for");
       slist.add("the");
       slist.add("The");
       slist.add("not");
       slist.add("with");
       slist.add("for");
       slist.add("from");
       slist.add("to");
       slist.add("-");
       slist.add("International");
       slist.add("Using");
       for(String s:slist){
          if(words_count.keySet().contains(s)){
           words_count.remove(s);
          }
       }
```

```
        }

        for(String s : words_count.keySet())
        {
          if(words_count.get(s)>3){
            //System.out.println(s +" = "+ words_count.get(s));
            mostFrequent = s;
            if(!commons.contains(mostFrequent)){
              commons.add(mostFrequent);
            }
          }
          /* Integer i = words_count.get(s);
          if(frequency == null)
              frequency = i;
          if(i > frequency)
          {
              frequency = i;
              mostFrequent = s;
              if(!commons.contains(mostFrequent)){
                commons.add(mostFrequent);
              }
          }*/
        }
      }
      //System.out.println("The word "+ mostFrequent +" occurred "+ frequency +" times");
  // }
   for(String mi:commons){
      System.out.println("First "+mi);
   }
}
exec = QueryExecutionFactory.sparqlService( "http://dblp.l3s.de/d2r/sparql", qs33.asQuery() );
results = ResultSetFactory.copyResults( exec.execSelect() );
if(results.hasNext()){
   while ( results.hasNext() ) {
      see = results.next().get( "title" ).toString();
      titles2.add(see);
      qresult.rowstring=see;
      qresultstitleArt.add(qresult);
      //System.out.println(see);
      qresult = new Qresult();
   }

  // for(int j=0;j<20;j++){
      Map<String,Integer> words_count = new HashMap<String,Integer>();
      Integer frequency2 = null;
```

```
String mostFrequent2 = null;
for(String st:titles2){
   String[] words = st.split("\\s+");
   for(int i=0;i<words.length;i++){
       String s = words[i];
       if(words_count.keySet().contains(s))
       {
           Integer count = words_count.get(s) + 1;
           words_count.put(s, count);
       }
       else{
           words_count.put(s, 1);
       }
   }

   for(int i=0;i<words.length;i++){
       List<String> slist = new ArrayList<String>();
       slist.add("A");
       slist.add("a");
       slist.add("An");
       slist.add("an");
       slist.add("as");
       slist.add("at");
       slist.add("and");
       slist.add("or");
       slist.add("of");
       slist.add("Of");
       slist.add("on");
       slist.add("On");
       slist.add("in");
       slist.add("for");
       slist.add("the");
       slist.add("The");
       slist.add("not");
       slist.add("with");
       slist.add("for");
       slist.add("from");
       slist.add("to");
       slist.add("-");
       slist.add("International");
       slist.add("Using");
       for(String s:slist){
          if(words_count.keySet().contains(s)){
           words_count.remove(s);
          }
       }
```

```
        }

        for(String s : words_count.keySet())
        {
          if(words_count.get(s)>3){
            //System.out.println(s +" = "+ words_count.get(s));
            mostFrequent2 = s;
            if(!commons2.contains(mostFrequent2)){
              commons2.add(mostFrequent2);
            }
          }
          /*Integer i = words_count.get(s);
          if(frequency2 == null)
            frequency2 = i;
          if(i > frequency2)
          {
            frequency2 = i;
            mostFrequent2 = s;
            if(!commons2.contains(mostFrequent2)){
              commons2.add(mostFrequent2);
            }
          }*/

        }
      }
      //System.out.println("The word "+ mostFrequent +" occurred "+ frequency +" times");
  // }
   for(String m:commons2){
      System.out.println("Second "+m);
   }
}
for(String c1:commons){
   for(String c2:commons2){
      if(c1.equals(c2)){
        qresult = new Qresult();
        qresult.rowstring=c1;
        nodelistbackgr.add(c1);
        qresultsback.add(qresult);
        System.out.println("Commons "+c1);
      }
   }
}
render5=true;
flag=1;
if(qresultsback.isEmpty()){
```

```
            flag=0;
        }
    }
    if(flag==0){                    //No Relationships where found
        see="No Relationship Found";
        qresult.rowstring=see;
        qresults.add(qresult);
        System.out.println(see);
        render5=false;
        render6=true;
    }


   // A simpler way of printing the results.
   // ResultSetFormatter.out( results );


}

public void reinit(){
    qresultsAuth=new ArrayList<>();
    qresults= new ArrayList<Qresult>();
    qresultsConf= new ArrayList<Qresult>();
    qresultsAuth= new ArrayList<Qresult>();
    qresultsArt= new ArrayList<Qresult>();
    qresultsArtJourn= new ArrayList<Qresult>();
    qresultsConfnew= new ArrayList<Qresult>();
    qresultstitleArt= new ArrayList<Qresult>();
    qresultstitle1Art= new ArrayList<Qresult>();
    qresultstitle2Art= new ArrayList<Qresult>();
    qresultsback = new ArrayList<Qresult>();
    render1=false;
    render2=false;
    render3=false;
    render4=true;
    render5=false;
    render6=false;
    nodelistconf = new ArrayList<String>();
    nodelistjrn = new ArrayList<String>();
    nodelistconferences= new ArrayList<String>();
    nodelistjournals = new ArrayList<String>();
    nodelistbackgr = new ArrayList<String>();
    titles1 = new ArrayList<String>();
    titles2 = new ArrayList<String>();
}
```

```
  /**************************** This Query returns the Details for the first Qurery ************************/
  public void coauthor(String coauthor){
     System.out.println(coauthor);

     ParameterizedSparqlString qsAuthors = new ParameterizedSparqlString(
          "prefix d2r: <http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2r-server/config.rdf#>\n"+
          "prefix swrc: <http://swrc.ontoware.org/ontology#>\n"+
          "prefix dcterms: <http://purl.org/dc/terms/>\n"+
          "prefix xsd: <http://www.w3.org/2001/XMLSchema#>\n"+
          "prefix dc: <http://purl.org/dc/elements/1.1/>\n"+
          "prefix map: <file:///home/diederich/d2r-server-0.3.2/dblp-mapping.n3#>\n"+
          "prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n"+
          "prefix foaf: <http://xmlns.com/foaf/0.1/>\n"+
          "prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>\n"+
          "prefix owl: <http://www.w3.org/2002/07/owl#>\n"+
          "\n" +
          "SELECT DISTINCT ?Author WHERE{\n"+
          "?paper dc:title ?title.\n"+
          "?paper dc:creator ?n.\n"+
          "?n foaf:name ?Author.\n"+
          "FILTER (?title = \""+coauthor+"\")\n"+
          "}\n");
     QueryExecution exec;
     exec = QueryExecutionFactory.sparqlService( "http://dblp.l3s.de/d2r/sparql", qsAuthors.asQuery() );

     resultsnew = ResultSetFactory.copyResults( exec.execSelect() );
     qresultsAuth=new ArrayList<>();
     while ( resultsnew.hasNext() ) {
          see = resultsnew.next().get( "Author" ).toString();
          qresultAuth.rowstringAuth=see;
          qresultsAuth.add(qresultAuth);
          qresultAuth = new Qresult();
     }
  }
  //private final MindmapNode root;

  private MindmapNode selectedNode;

  /*public Bean() {
     root = new DefaultMindmapNode(name, "Google WebSite", "FFCC00", false);

     MindmapNode ips = new DefaultMindmapNode("IPs", "IP Numbers", "6e9ebf", true);
     MindmapNode ns = new DefaultMindmapNode("NS(s)", "Namespaces", "6e9ebf", true);
     MindmapNode malware = new DefaultMindmapNode("Malware", "Malicious Software", "6e9ebf", true);
```

```
    root.addNode(ips);
    root.addNode(ns);
    root.addNode(malware);
}*/

public MindmapNode getRoot() {
    return root;
}

public MindmapNode getSelectedNode() {
    return selectedNode;
}
public void setSelectedNode(MindmapNode selectedNode) {
    this.selectedNode = selectedNode;
}

public void onNodeSelect(SelectEvent event) {
    MindmapNode node = (MindmapNode) event.getObject();

    //populate if not already loaded
    if(node.getChildren().isEmpty()) {
        Object label = node.getLabel();

        if(label.equals(coauthor)) {
            if(!nodelistconf.isEmpty()){
                for(String c:nodelistconf) {
                    node.addNode(new DefaultMindmapNode(c, "Publication: "+c, "fce24f", false));
                }
            }
            if(!nodelistjrn.isEmpty()){
                for(String j:nodelistjrn) {
                    node.addNode(new DefaultMindmapNode(j, "Publication: "+j, "3399ff", false));
                }
            }
            if(!nodelistconferences.isEmpty()){
                for(String con:nodelistconferences) {
                    node.addNode(new DefaultMindmapNode(con, "Conference: "+con, "82c542", false));
                }
            }
            if(!nodelistjournals.isEmpty()){
                for(String jou:nodelistjournals) {
                    node.addNode(new DefaultMindmapNode(jou, "Journal: "+jou, "3399ff", false));
                }
            }
            if(!nodelistbackgr.isEmpty()){
```

```
            for(String bgr:nodelistbackgr) {
               node.addNode(new DefaultMindmapNode(bgr, "Background: "+bgr, "82c542", false));
            }
         }
      }
   }

   public void onNodeDblselect(SelectEvent event) {
      this.selectedNode = (MindmapNode) event.getObject();
   }
}
```

Qresult.java

```
public class Qresult {
   public String rowstring;
   public String rowstringnew;
   public String rowstringAuth;
   public String rowstringArt;
   public String rowstringArtJourn;
   public String rowstringnew2;

   public String getRowstring(){
      return rowstring;
   }

   public void setRowstring(String rowstring){
      this.rowstring=rowstring;
   }

   public String getRowstringnew(){
      return rowstringnew;
   }

   public void setRowstringnew(String rowstringnew){
      this.rowstringnew=rowstringnew;
   }

   public String getRowstringAuth(){
      return rowstringAuth;
   }

   public void setRowstringAuth(String rowstringAuth){
      this.rowstringAuth=rowstringAuth;
   }
```

```
    public String getRowstringArt(){
        return rowstringArt;
    }

    public void setRowstringArt(String rowstringArt){
        this.rowstringArt=rowstringArt;
    }

    public String getRowstringArtJourn(){
        return rowstringArtJourn;
    }

    public void setRowstringArtJourn(String rowstringArtJourn){
        this.rowstringArtJourn=rowstringArtJourn;
    }

    public String getRowstringnew2(){
        return rowstringnew2;
    }

    public void setRowstringnew2(String rowstringnew2){
        this.rowstringnew2=rowstringnew2;
    }
}
```